IHW

Attorney Docket No. 9342-98 PATENT

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re: Kerimovska et al.
Serial No.: 10/539,238
Filed: April 10, 2006
Confirmation No.: 9239
Group Art Unit: 2626
Examiner: Jakieda R. Jackson
For: **DEVICE FOR GENERATING SPEECH, APPARATUS CONNECTABLE TO OR INCORPORATING SUCH A DEVICE, AND COMPUTER PROGRAM PRODUCT THEREFOR**

Date: October 11, 2007

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

SUBMITTAL OF PRIORITY DOCUMENTS

Sir:

Pursuant to Examiner Jackson's request, enclosed is a certified copy of European priority Patent Application No. 02445177.5 filed on December 16, 2002 and a certified copy of European priority Patent Application No. 03011580.2 filed on May 22, 2003.

If any extension of time for the accompanying response or submission is required, Applicants request that this be considered a petition therefor. No fee is believed due, however, the Commissioner is hereby authorized to charge any deficiency, or credit any refund, to our Deposit Account No. 50-0220.

Respectfully submitted,

Rohan G. Sabapathypillai
Registration No. 51,074

Customer No. 54414
Myers Bigel Sibley & Sajovec
PO Box 37428
Raleigh NC 27627
Tel (919) 854-1400
Fax (919) 854-1401

Certificate of Mailing under 37 CFR 1.8

I hereby certify that this correspondence is being deposited with the United States Postal Service with sufficient postage as first class mail in an envelope addressed to: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450, on October 11, 2007.

Betty-Lou Rosser

LUND INSTITUTE
OF TECHNOLOGY
Lund University

Sony Ericsson

Text-to-Speech For Mobile Application

Master of Science Thesis

Anna Tomasson & Nercivan Kerimovska
in cooperation with
Sony Ericsson Mobile Communications AB

December 2002

Abstract

Text-to-speech conversion is a feature that is of interest in many different areas and applications, one of the more interesting is the use in mobile phones. Today mobile phones are used by everyone and a feature like this can be an important aid especially for the visual impaired and for users who need to focus on other things while using a phone.

To show some possibilities a demonstrator that implements text-to-speech has been designed. With this attached to the phones system connector the user can have SMS and menu labels read.

The text-to-speech conversion is done in hardware with a text-to-speech circuit. The highlighted menu label or the SMS is sent to a microcontroller via the system connector. The texts are received as ASCII characters and these are forwarded to the text-to-speech circuit by the microcontroller. The text-to-speech circuit converts the characters to audio signals and sends them to a loudspeaker.

Contents

Abbreviations

ASCII	American Standard Code for Information Interchange
ASIC	Application-Specific Integrated Circuit
AT	Attention
AVR	Manufacturer
CAD	Computer Aided Design
CFMS	Command From Mobile Station
CMOS	Complementary Metal-Oxide Semiconductor
CPOL	Clock Polarity
CS	Clock Select
CTMS	Command To Mobile Station
DFMS	Data From Mobile Station
DIL	Dual In-Line
DTMS	Data To Mobile Station
EEPROM	Electrically Erasable Programmable Read-Only Memory
GPS	Global Positioning System
I/O	Input/Output
ISP	In-System Programming
MIDI	Musical Instrument Digital Interface
MISO	Master In Slave Out
MLS	Multi-Level memory Storage

MOSI	Master Out Save In
MSTR	Master/Slave select
PBA	Printed Board Assembly
PC	Personal Computer
PCB	Printed Circuit Board
RISC	Reduced Instruction Set Computer
RS-232	Standard for serial communication
SCLK	Serial Clock
SDI	Serial Data In
SDO	Serial Data Out
SMS	Short Message Service
SPCR	SPI Control Register
SPDR	SPI I/O Data Register
SPE	SPI Enable
SPI	Serial Peripheral Interface
SPIE	SPI Interrupt Enable
SPIF	SPI Interrupt Flag
SRAM	Static Random Access Memory
SS	Slave Select
TQFP	Thin Quad Flat Pack
TSOP	Thin Small Outline Packages
TTS	Text-To-Speech
UART	Universal Asynchronous Receiver and Transmitter

UBRR	UART Baud Rate Register
UDR	UART Data Register
UV	Ultra Violet
WAP	Wireless Application Protocol

1 Introduction

Mobile phones have been a part of our lives for over two decades and have experienced an explosive technical development. The main function is still, of course, to set up phone calls but today there are a lot of other possibilities. The technical progress has made it possible to make the phones cheaper and smaller but more usable. Mobile phones have become user-friendlier with polyphonic sounds and larger color displays. They are today also developed for all kinds of different environments.

The phones have been developed with focus to fit different kinds of user, for example the possibility for businessmen to make conference calls and teenagers to play games. There is still one category that is excluded; those who are visual impaired. A larger display with different colors is not enough. With help of other senses they might be able to use a phone like everybody else. The solution is to make the phone talk. Is that possible?

An ordinary service like SMS has been completely ruled out since it has been impossible to read them. A mobile phone adapted for this group must among other things help the user by reading SMS and maybe most important of all by reading the menus to help locate while browsing the menu system.

A solution like this is also a benefit to other phone users. Especially when the phone is not in focus, for instance while driving. With this aid the driver can pay attention to the road and not to the phone.

1.1 The Goal and How to achieve it

The goal with this master's thesis was to develop a prototype for text-to-speech (TTS) conversion that should convert menus and SMSs into spoken text. The prototype should be attached to the mobile phone T68is system connector as an accessory. This requires that the prototype is of proportional size, i.e. not much larger than the mobile phone itself and that it is not ungainly. The goal was clear, but how was it approached?

The idea was to use an already existing TTS circuit and combine it with a microcontroller. The block diagram in Figure 2.1 gives an overview of the prototype.

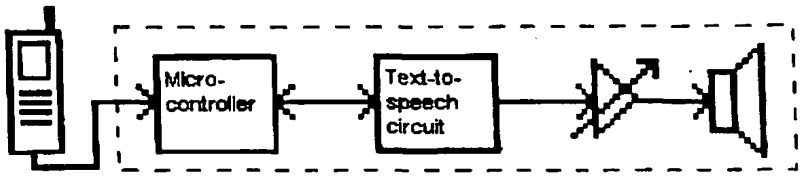


Figure 2.1 Block diagram over the main blocks i.e. phone, microcontroller, TTS circuit, amplifier and loudspeaker.

The parts within the dashed square are intended to represent the accessory. The idea was to let the microcontroller get the text to be converted from the mobile phone and pass it through to the TTS circuit. The TTS circuit converts the text to audio signals and sends them via an amplifier to a loudspeaker. The phone was supposed to communicate with the accessory with attention (AT) commands. AT commands and certain protocols are used for communication between the phone and accessories connected to the phone system connector.

1.2 Report Outline

The outline of this report is as follows:

Chapter 1: In this chapter an introduction to the report is given.

Chapter 2: This chapter describes the underlying research and contains the theory about the circuits to be used. Furthermore it contains a study of the software design and the interfaces needed.

Chapter 3: This chapter describes the different parts of the implementation process; wiring, etching and mounting of components and programming of the phone and microcontroller.

Chapter 4: In this chapter the tools used during the development of the accessory are presented.

Chapter 5: The result of the thesis is discussed and the limitations of the accessory are described in this chapter. It also contains a part for user instructions.

Chapter 6: This chapter discusses other possibilities with this application and different ways to implement it.

2 Research

To achieve the goal a proper research must be done; an investigation of what hardware to be used and how the software should be designed. The interfaces between the circuits and how they should be connected must also be considered before a final decision of the shape of the prototype can be made. An important step in this procedure is a careful study of data sheets and application notes.

The development process was planned to consist of six steps:

1. **Research** – pre-study of how the development will proceed, decision of which circuits to be used and study of documentation/data sheets.
2. **Wired prototype** – a primer coupling is wired for basic error detection and to check the expected hardware behavior.
3. **The microcontroller software** – initialization of the circuits and implementation of the interfaces and the functionality required.
4. **The phone software** – implementation of needed software in the already existing phone software.
5. **Circuit board** – computer aided design (CAD) of the printed circuit board (PCB), etching and mounting of components.
6. **Flashing** – programming both the phone and the microcontroller with the software mentioned in steps 3 and 4.

2.1 The Hardware

The Atmel AVR 90S8515 was the pre-defined microcontroller to be used in this project and on the basis of this the other circuits were chosen.

Most of the TTS circuit manufacturers demonstrate speech samples of their circuits on the Internet. The Winbond WTS701 was selected among a number of different, existing TTS circuits on the market, since it has the best speech quality, is a single-chip solution and seemed to be relatively easy to mount and use. This circuit has a differential output designed to drive an 8 Ω speaker and apart from this also a tele socket for a headphone or an external loudspeaker was suggested. The tele socket works in such a way that when an external loudspeaker is connected to it, the built-in loudspeaker is disconnected. To get a higher volume level an amplifier is also needed. Figure 2.1 shows the interaction between the different hardware blocks.

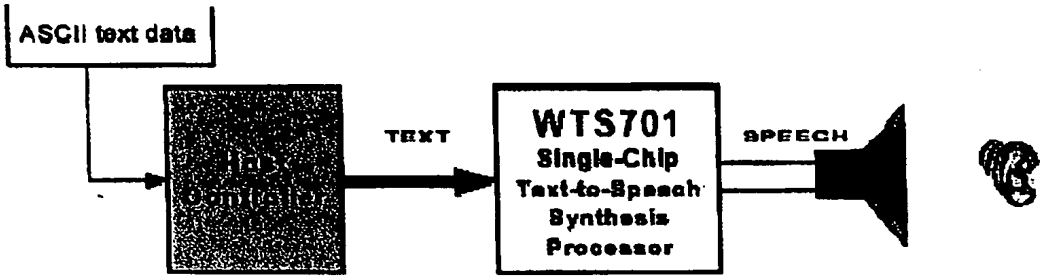


Figure 2.1 An overview of the hardware coupling [9].

For customization it should be possible to change the volume and therefore buttons are needed.

The accessory must also have a system connector to be able to connect to the phone.

If possible, it would be convenient to let the phone supply the accessory with power. According to the datasheets of the parts planned to be used, the maximum voltage level is limited by the data voltage level between the phone and the accessory to +2.7 V. This level is below the voltage (3.3-4.1 V) that the phone may deliver to an external device. It is therefore possible to let the phone supply the accessory. To convert the supply voltage to +2.7 V, a voltage converter is necessary.

2.1.1 The Microcontroller

The AT90S8515 is a low-power CMOS 8-bit microcontroller based on the AVR RISC architecture. The most important features of the microcontroller are as follows:

- 8 K bytes of In-System Programmable Flash
- 32 general-purpose I/O lines
- 32 general-purpose working registers
- internal and external interrupts
- a programmable serial Universal Asynchronous Receiver and Transmitter, UART
- a port for the Serial Peripheral Interface, SPI, see chapter 2.3

The On-chip In-system Programmable Flash allows the program memory to be reprogrammed In-System through SPI. This means that it is possible to flash the microcontroller with new software without removing it from the circuit board, see chapter 4.5.

The microcontrollers operating voltages are +2.7-6 V and its speed grades 0-4 MHz. The power consumption at 4 MHz, +3 V, 25°C is 3.0 mA in active mode and less than 1 µA in power-down mode.

AT90S8515 has four different ports named A, B, C and D that consists of 8 pins each and serves as 8-bit bi-directional I/O ports. Some of the pins have other functionalities that are enabled by writing to the corresponding register. The registers control the behavior of the microcontroller and by writing to these the functionality of the microcontroller is customized. Parts of port A and C can be used for communication with an external memory. Some of the port B pins may serve as the SPI interface and two pins on port D can be used for UART communication.

The microcontroller has a corresponding real-time emulator, ATICE200, whose adapter is pin compatible and can be used for developing a wired prototype. The emulator simulates and behaves like the microcontroller but is controlled by a computer. It makes it possible to follow each command and it shows how the different ports, pin values and registers change during execution. It is also possible to single-step i.e. execute one line of the code at a time to see if the behavior is as expected, see chapter 4.4. This is a great advantage that aims for easier debugging, and therefore planned to be used in this project. Since a wired prototype, which contains hole mounted circuits, will not have an acceptable size also a prototype with surface mounted circuits must be developed.

2.1.2 The Text-to-Speech Circuit

The TTS circuit, Winbond WTS701, is a high quality, fully integrated and single-chip solution that is ideal for this application, mainly because it supports SMS and has a general modifiable abbreviation list. The supported languages today are U.S. English and Mandarin (Chinese dialect) but other languages are in development or in planning. To change language it should be possible to program through the SPI port allowing the user to download different languages and speaker databases. Today, changing language requires that the chip is manually exchanged.

The U.S. English TTS circuit accepts ASCII input via the SPI port, see chapter 2.3, and converts it into spoken audio via an analogue output. It integrates a text processor, a smoothing filter and multi-level memory storage (MLS) array on a single-chip. Voice and audio signals are stored in the memory in their natural, uncompressed form which provides a good voice reproduction quality.

TTS conversion is achieved by dividing the incoming text into phonetic representations that is then mapped to the array that contains naturally spoken word parts. The synthesis algorithm tries to use the largest possible word unit in the appropriate context to maximize natural sounding. When all parts of a word have been mapped the phonetic representations are placed in a queue. These representations are then sent to the analogue output.

The TTS circuit has an internal input buffer that can hold up to 256 characters and can therefore receive an entire SMS consisting of 160 characters. This means that no extra memory is needed.

The TTS is a low power consumer; with the supply voltage +2.7 V it requires an operation current equal to 35 mA typical and at stand-by less than 1 μA.

2.1.2.1 The Text-to-Speech Mechanism

The TTS component of the system consists of three principal blocks:

- Text normalization
- Word to phoneme conversion
- Phoneme mapping

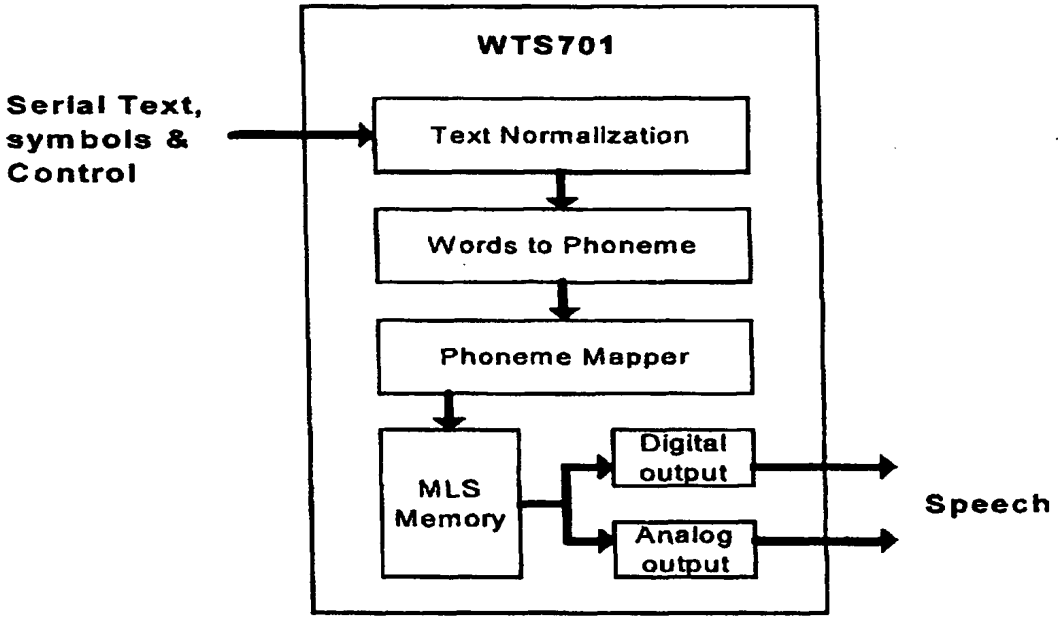


Figure 2.2 TTS system process flow. The digital output represents a CODEC interface, while the analogue output contains a D/A converter. The CODEC interface is not used in this project [10].

2.1.2.2 Text Normalization

This is the process of translating the incoming text to pronounceable words. It expands abbreviations and translates numeric strings to spoken words.

The abbreviation list can be modified. This enables flexibility of adding abbreviations specifically for the text, either by the developer or by the end user to customize the product. Even the unique characters of SMS are supported, meaning that icons such as smilies will be replaced by its true meaning. This means that an SMS containing abbreviations and icons will be correctly recited.

2.1.2.3 Words-to-Phoneme Mapping

When the text has been translated into pronounceable words, the system must decide how to pronounce them. This is not an easy task since every language is made of a complicated set of rules for pronouncing. To handle this task the system works with a combination of rule based processing and exception processing. An example of a word that is pronounced differently depending on the rest of the sentence is 'read'. It can either be pronounced *red* or *reed*.

2.1.2.4 Phoneme Mapping

No dictionary is unlimited, which means that the system must divide words into sub-words that it recognizes from the stored list of words in the internal memory. The dividing or splitting must be done at appropriate phonetic boundaries to achieve high quality concatenation. Once a sub-word unit is determined, the inventory is searched to determine if a match is present. A matching weight is assigned to each match depending on how closely the phonetic context matches. Each sub-word has a left and right side context to match as well as the phoneme string itself. If no suitable match is found in the inventory, the sub-word is further split in a tree-like manner until a match is found. The splitting tree is processed from left to right and each time a successful match occurs, the address and duration of the match in the corpus is placed in a queue of phonetic parts to be played out in the audio interface.

2.1.3 The Amplifier

The LM4894MM is a fully differential audio power amplifier with two amplifying blocks, capable of delivering 1 W of continuous average power to an 8 Ω bridge-connected load. The gain in each block is set with two resistors, R_F and R_I, for each input, Equation 2.1.

$$Gain = -\frac{R_F}{R_I} \tag{2.1}$$

It is important to match the input resistors, R_I, and the feedback resistors, R_F, to each other in order to get the right amplification. Since the amplifier is bridge-coupled with two outputs, the total gain will be twice the gain in Equation 2.1, Equation 2.2.

$$Gain_{tot} = Gain \cdot 2 \quad (2.2)$$

To achieve a $Gain_{tot}$ of 2, R_F must be twice the value of R_I. In this project R_I is set to 10 kΩ and R_F is then set to 22 kΩ.

To save power it is possible to set the amplifier in shutdown mode. This is done by toggling its Shutdown Select pin to the same state as its Shutdown Mode pin, [redacted]

2.1.4 The Voltage Regulator

The MAX8863 is a low-dropout linear regulator that operates from +2.5 V to +6.5 V input voltage and delivers up to 100 mA. The output voltage is either preset or adjusted between +1.25 V to +6.5 V, by two external resistors as a voltage divider, Equation 2.3.

$$V_{out} = V_{set}\left(1 + \frac{R_1}{R_2}\right) \quad (2.3)$$

V_{set} is 1.25 V and R_2 is set to 100 kΩ to optimize power consumption. The divider consists of the two resistors R_1 and R_2. To achieve a supply voltage of +2.7 V R_1 is calculated to 116 kΩ. To be sure that the voltage level is above +2.7 V, R_1 is set to 120 kΩ.

2.1.5 The Buttons

Every press on one of the buttons should make the microcontroller either increase or decrease the volume. The microcontroller can easily detect a press by having the buttons connected to its interrupt pins (INT0 and INT1) and letting them generate interrupts.

2.1.6 The System Connector

To be able to connect the accessory to the phone, a system connector is needed. The system connector interface consists of audio signals, two serial channels, power leads and the analogue and digital ground leads. For an overview of the connector and its pins, see Figure 2.3.

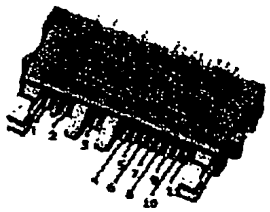


Figure 2.3 Full System Connector, accessory side [5].

2.2 The Software

The microcontroller is the circuit that will control the entire accessory and the communication with the phone. It must therefore be programmed to achieve the wanted behavior. Since the microcontroller can not control the phone, no accessory is allowed to do that, programming in the phone is also required. Both the code in the microcontroller and the code in the phone is written in C.

The initial plan was to use AT commands to let the accessory communicate with the phone by sending requests, for the text strings, on the system connector Command to Mobile Station (CTMS) pin. The Command from Mobile Station (CFMS) pin is used for receiving commands from the phone. All data transfer goes via the Data to Mobile Station (DTMS) and Data from Mobile Station (DFMS) pins. This proved to be very complicated since no AT commands for sending text from the phone exists. To circumvent the problem it is possible to let the phone print the data to the system bus in the same way as it does when it is being logged. To log the phone, a special plug is connected between the phone system connector and a computer. The plug enables the phone to put data on the system bus. When the phone is connected in this way, only one serial channel is used and all data on the system bus is available on the CFMS pin. To place data on the bus an ordinary *printf()* command can be used.

2.2.1 The Microcontroller Software

To be able to control the accessory, the microcontroller must be initiated and customized for this purpose. This is done by setting its registers, defining which pins to be inputs/outputs, and enabling the alternative functions.

The microcontroller software must also initiate the TTS circuit and include a set of communication commands and functions for the SPI and UART interfaces, see chapter 2.3. It must also handle and react on possible interrupts.

2.2.2 The Phone Software

The phone must support the accessory in the way that it must send the data to it. To put text strings on the system bus, a few lines of code must be added to affected modules in the general T68i code. The main purpose is to extract the text string from the text id and send it to the accessory with a *printf()* command. All text strings are stored in a list and the text id is a pointer used to point out the different text strings.

2.3 The Interfaces

The different blocks in the system need the right interfaces to communicate properly with each other. The interface between the phone

and the microcontroller consists of a UART, chapter 2.3.1, while the microcontroller and the TTS circuit communicate via SPI, see chapter 2.3.2.

The data (ASCII characters) sent from the phone to the accessory is intended to be received and handled by the microcontrollers Universal Asynchronous Receiver and Transmitter, UART, and then to be sent to the TTS circuit. The communication between the TTS circuit and the microcontroller is supposed to be handled by the Serial Peripheral Interface, SPI. The data flow is illustrated in Figure 2.4.

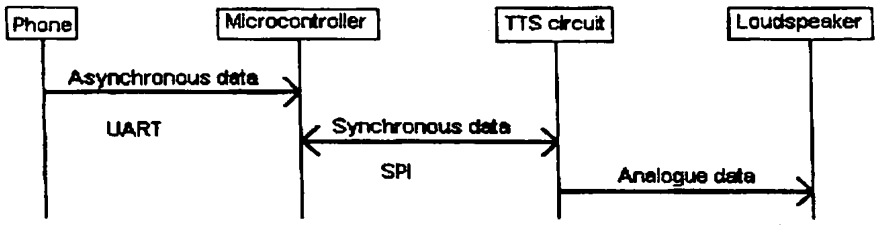


Figure 2.4 Data flow diagram.

2.3.1 The UART

The microcontroller has a full duplex UART with the main features:

- Baud rate generator that can generate a large number of baud rates
- High baud rates at low crystal frequencies
- 8 or 9 bits data
- Noise filtering
- Overrun detection
- Framing error detection
- False start bit detection
- Interrupts

2.3.1.1 Data Transmitter

Data transmission is initiated by writing the data to be transmitted to the UART I/O Data Register (UDR). The data is then transferred from UDR to the transmitter shift register and shifted out on the UART output line (TX pin).

Since AT commands was decided not to be used, there is no need to transfer data to the phone and the data transmitter is therefore of no interest in this application.

2.3.1.2 Data Receiver

The receiver samples the incoming signal on the UART input line (RX pin) at a frequency 16 times the baud rate. The baud rate can be set in the

UART BAUD Rate Register (UBRR), see Appendix C. Every bit received will be sampled 16 times and sample 8, 9 and 10 will decide the logical value of the bit, see Figure 2.5.

Figure 2.5 Sampling data [8].

An incoming sample of logical '0' will be interpreted as the falling edge of a start bit. After a valid start bit, sampling of the data bits is performed and finally a stop bit is received. Whether or not a valid stop bit is detected, the data is transferred to UDR, a flag is set and the UART Receive Complete interrupt is executed. UDR is in fact two physically separate registers, one for transmitted data and one for received data.

2.3.2 The SPI

The SPI is a serial interface developed by Motorola and is used primarily for synchronous serial communication of host processor and peripheral circuits. The standard SPI system is made of one master and one, or more, slaves. A device may be in master or in slave mode, but not in both at the same time. The master device provides the clock signal and determines the state of the chip select lines, i.e. it activates the slave it wants to communicate with. This means there is one master while the number of slaves is only limited by the number of chip selects. Since an ordinary I/O pin may be used as chip select, it is the master that limits the number of slaves. In this application the microcontroller is the master and the TTS circuit is the only slave.

The basic principle behind the SPI is an ordinary shift register. Command codes as well as data are serially transferred, clocked into the shift register and then internally available for parallel processing. The length of the shift register is not fixed and may differ from device to device, but normally there is space for 8 bits.

The SPI requires two control lines, Chip Select (CS) and Serial Clock (SCLK), and two data lines, Serial Data In (SDI) and Serial Data Out (SDO). Motorola names these lines Master-Out-Slave-In (MOSI) and Master-In-Slave-Out (MISO). The chip select line is named Slave-Select (SS). The majorities of the SPI devices provide these four lines, see Figure 2.6.

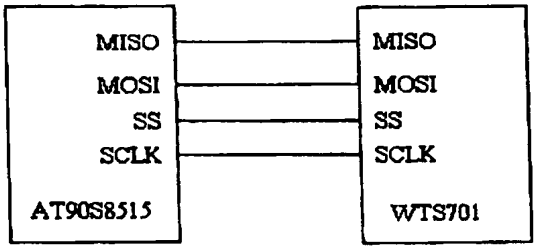
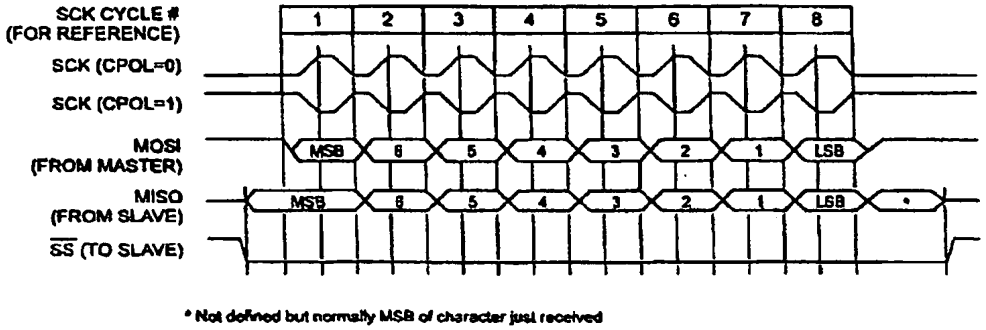


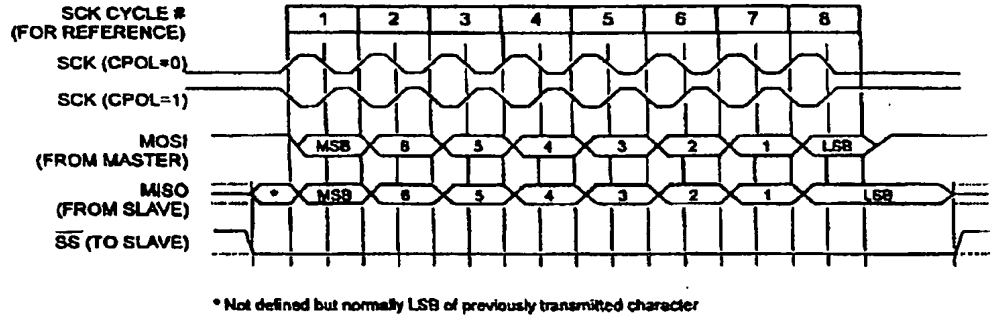
Figure 2.6 The different SPI lines.

Because there is no official specification what SPI is, it is necessary to consult the data sheets to see the specific demands of clock synchronization i.e. the permitted clock frequencies and the type of valid transitions. In practice four modes are used for transitions and with these the signals are adjusted so they can be understood and interpreted correctly by the receiver. These four modes are the combination of Clock Polarity (CPOL) and Clock Phase (CPHA). It is important that these agree both with master and slave, see Figures 2.7 and 2.8.



* Not defined but normally MSB of character just received

Figure 2.7 SPI transfer format with CPHA=0 [8].



* Not defined but normally LSB of previously transmitted character

Figure 2.8 SPI transfer format with CPHA=1 [8].

2.3.2.1 The Communication Protocol

The microcontroller can control the TTS circuit by sending commands via the SPI interface. Transactions always start by sending a command. The command consists of a command word and a command byte. At the same time the command is shifted out on the MOSI line, the TTS circuit shifts out its status register on the MISO line, Figure 2.9.

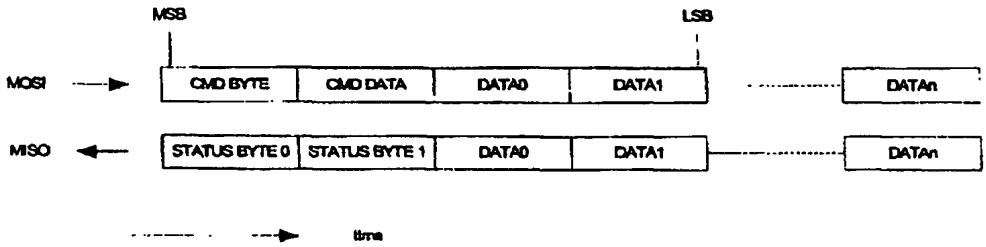


Figure 2.9 The format for a SPI transaction [10].

There are four different types of commands, depending on the interaction with the TTS circuit. Class 0 commands are commands that will execute irrespective of the state of the circuit. A typical class 0 command is the Start Up command. Class 1 commands require interpretation by the internal firmware, and an example of this is the Set Volume command. Class 2 commands, for example the Convert Text command, have associated data and finally class 3 commands have data to return to the host. Class 3 commands are used for modifying the abbreviation list and are not used in this project.

2.3.2.2 The Communication Signals

A SPI transaction is initiated when the master writes data to its SPI data register, the SPDR register. This register contains 8 bits. Writing to this register starts the SPI clock generator, SCLK. The only time this clock is active is during an SPI transaction. The slave responds to a command when the CS is low and addressed by an active low signal on its SS pin. Since this system only contains one slave, the TTS circuit, the SS pin of the master, i.e. the microcontroller, is not used. Under this condition, the TTS circuit accepts data on the MOSI input, which is clocked in on rising edges of the SCLK signal. The data written in SPDR is shifted out on the master MOSI line and in on the MOSI line of the slave. After shifting one byte the SPI generator stops, sets the end-of-transmission flag (SPIF) and an end-of-transmission interrupt is executed. This interrupt is enabled by setting the SPIE bit in the SPCR register. An SPI transaction is finished when SS is returned to the high condition. The two shift registers in the master and the slave can be considered as one distributed 16-bit circular shift register, Figure 2.10.

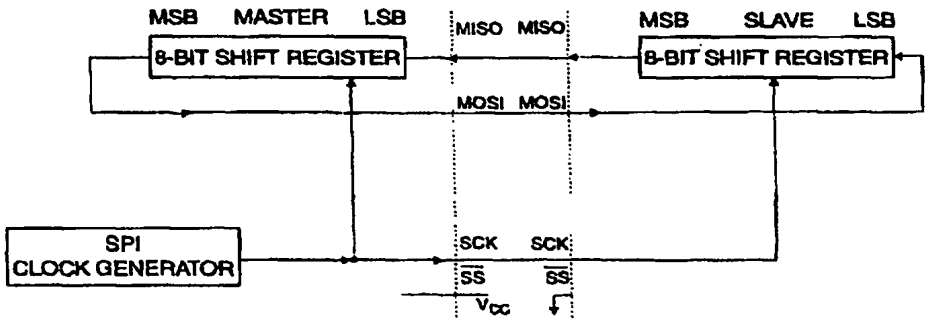


Figure 2.10 The SPI master and slave shift registers [8].

When data is shifted from the master to the slave, data is simultaneously shifted in the opposite direction. This means that during one shift cycle, data in the master and the slave are interchanged. The system is single-buffered in the transmit direction and double-buffered in the receive direction. This means that bytes to be transmitted cannot be written to the SPI data register before the entire shift cycle is completed. When receiving data, a received byte must be read from the SPI data register before the next byte has been completely shifted in. Otherwise the first byte is lost.

To enable the SPI on the microcontroller the SPE bit in the SPI control register SPCR must be set and to select master mode the MSTR bit in the same register is set. See Appendix C for further information of the register settings.

3 Implementation

With the starting point of the parts decided to be used, a wired prototype was created. The reason for doing this is that it will simplify error tracing. This makes it easier to measure the signals with an oscilloscope and to detect errors in the coupling. It is also easier to modify the original train of thought. When the wired prototype was working properly the final circuit board was created.

The circuits on the wired prototype are all hole mounted circuits, except for the TTS circuit that requires a special socket. The components on the final circuit board are all surface mounted.

3.1 The Hardware

The hardware used in the wired prototype is the same as used in the final circuit board except for the microcontroller. The microcontroller used on the wired prototype is of different shape and pin count than the one used on the circuit board.

3.1.1 The Circuits

A description of how the circuits are connected according to the circuit diagram follows below. The final circuit diagram is shown in Appendix A.

3.1.1.1 The Microcontroller

Despite of the differences in the shapes between the microcontroller on the wired prototype and the one on the circuit board, the pins are connected in the same way and as follows:

Port A: the pins PA0-PA2 are used for control communication between the microcontroller and the TTS circuit. PA0, configured as an output, is connected to the SS pin on the TTS circuit and is used for initiating SPI transmission. PA1 is an input connected to the RB pin on the TTS circuit and indicates if the TTS circuit is ready to accept commands or not. The pin PA2 is used for resetting the TTS circuit and is therefore connected to its RESET pin. PA3 is connected to the SD SEL pin on the amplifier and is used for turning the amplifier on.

Port B: the pins PB5-PB7 are used for SPI and are connected to the corresponding pins either on the TTS circuit or to the flash socket used by AVRISP.

Port C: since no external memory is needed, port C is not used.

Port D: pin PD0 is the UART input line, used for communication with the phone, and is connected to the CFMS pin on the system connector. PD2 and PD3 serve as external interrupts and are used for volume control.

Pin XTAL1 and XTAL2 are the input to and output from the inverting oscillator amplifier and to the internal clock operating circuit. A crystal with a frequency of 4 MHz, according to the datasheets, is connected as close to the microcontroller as possible.

The RESET pin on the microcontroller is connected either to the AVRISP or supply voltage via the flash socket.

GND is connected to ground and VCC to supply voltage i.e. +2.7 V.

The remaining pins are not used in this application.

When both the hardware and the software behaved as expected, the emulator was changed to a real hole-mounted microcontroller. Since the emulator has a pin count of 40 pins, it must be exchanged to a microcontroller of the same shape. For this task, a microcontroller with a 40 pin DIL package was used, see Figure 3.1a. This is not the same package intended to be used in the final circuit board; for this a 44 pin TQFP was used, Figure 3.1b. The microcontrollers have exactly the same behavior; the four extra pins are of no use, i.e. not connected.

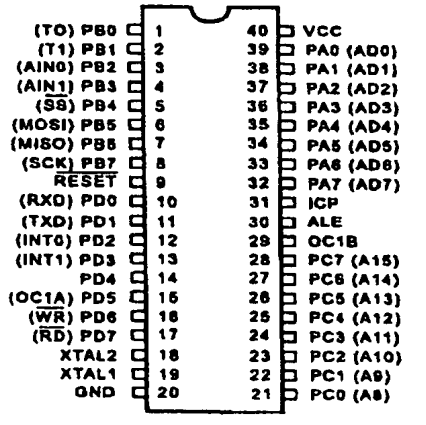


Figure 3.1a Microcontroller as DIL.

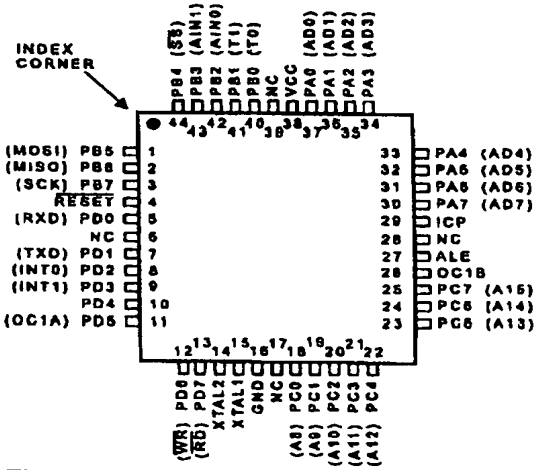


Figure 3.1b Microcontroller as TQFP.

3.1.1.2 The Text-to-Speech Circuit

The TTS circuit used in the prototype is exactly the same that is used in the final circuit board, which means that it is surface-mounted. To be able to connect the circuit to the wired board, a special socket must be used.

The socket converts a surface mounted circuit to a hole-mounted one. The TTS circuit has 56 pins, which is an unusual pin count, and to be able to connect the socket a special circuit board must be created. The circuit board translates the pin configuration to a shape that fits the prototype board.

Apart from the pins connected to the microcontroller mentioned in chapter 3.1.1, the rest of the pins are connected as follows:

The pins VSSA, VSSD are connected directly to ground while the pins AUXIN (according to data sheet) and CS (sets the TTS circuit to slave) are connected to ground via pull-down resistors.

The pins VCCD and VCCA are connected to supply voltage.

The pins XTAL1 and XTAL2 are connected to a crystal with a frequency of 24.576 MHz to ensure correct functionality. The crystal has to be mounted as close to the TTS circuit as possible for correct behavior.

The SP+ and SP- pins are connected to the amplifier IN+ and IN- pins.

3.1.1.3 The Amplifier

Since both the mounted loudspeaker and the headphone needed amplification it seemed natural to connect the amplifier directly after the TTS circuit. The IN+ and IN- pins are connected as described in chapter 3.1.2 and the rest of the pins are connected as follows:

In this application the SD MODE pin is connected to supply voltage. If the SD SEL pin, connected to PA3 on the microcontroller as described in chapter 3.1.1, is held low the amplifier will be switched on all the time.

The pin VDD is connected to the system connectors supply voltage pin since the supply voltage +2.7 V is not enough according to chapter 2.1.3. The GND and BYPASS pins are connected to ground.

The VO1 and VO2 pins are connected to the tele socket. These pins are also connected to the IN+ respective IN- through feedback resistors.

3.1.1.4 The Voltage Regulator

The pins IN and SHDN are connected to the supply voltage pin on the system connector, and GND is connected to ground. The OUT and SET pins are connected to a voltage divider that sets the output voltage level used as the supply voltage for the rest of the accessory.

3.1.1.5 The Buttons

The buttons are connected to PD2 and PD3 i.e. the external interrupt pins on the microcontroller, chapter 3.1.1. Since these pins have internal pull-up resistors, a press on any one of the buttons will generate a low pulse on the corresponding pin.

3.1.1.6 The System Connector

The data to be sent from the phone is put on CFMS and this pin is connected to the UART input line on the microcontroller, PD0. The voltage supply pin is connected to the input voltage pin on the voltage regulator and the ground pin connects the phone ground to the accessory ground.

3.1.2 The Circuit Board

The final step was to develop the accessory in its true shape. This was done in the following steps:

1. A circuit diagram, showing the couplings between the circuits, was drawn, see Appendix A.
2. A placement drawing, showing the physical placement of the circuits, and the template for the Printed Circuit Board, PCB, was created, see Appendix B.
3. The PCB was etched with the pattern created in step 2.
4. The Printed Board Assembly, PBA, was created by mounting the components on the PCB.

The circuit diagram (step 1 above), the placement drawing and the template (step 2) were created in Easy-PC Number One System which is an electrical CAD program, see chapter 4.1. To make the accessory as small as possible, the PCB and the PBA were made double sided i.e. they have components on both sides, see Figures 3.2 and 3.3.

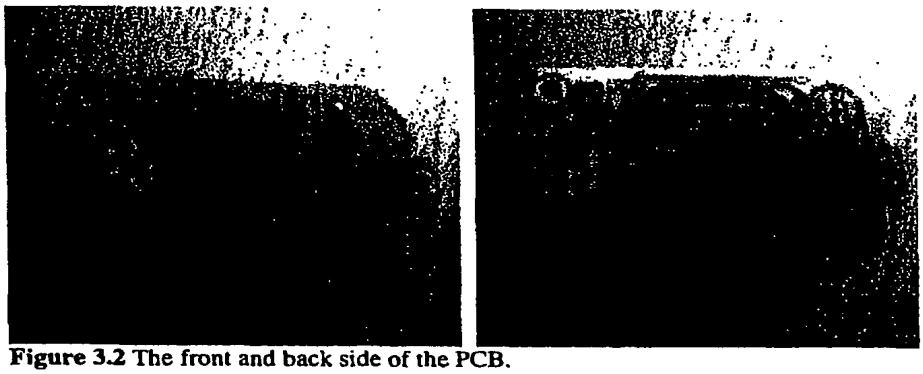


Figure 3.2 The front and back side of the PCB.

Figure 3.3 The front and back side of the PBA.

3.1.2.1 Etching and Soldering

The template created in Easy-PC, Appendix B, was printed on transparent plastic and placed on a photo resist copperplate. The plate must then be exposed to UV-light to make the photo resist stick to the copper. Following this the plate had to be dipped in a developer and thereafter placed in an etching bath. After the bath the plate was washed with water and finally wiped with acetone.

When making the PCB board for the accessory, the plate was exposed for approximately 6 minutes and was etched for about 10 minutes. It is not possible to define a precise time in this process. It is a work of feeling. The different steps in the etching process can be seen in the Figures 3.4-3.10.

Figure 3.4 The copper plate.

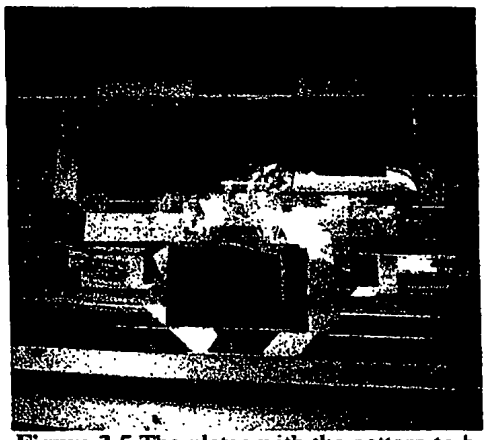


Figure 3.5 The plates with the pattern to be exposed.

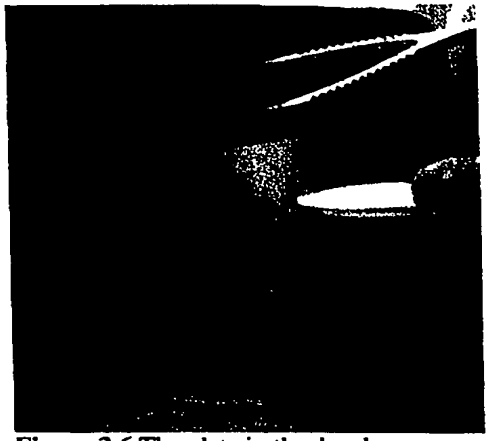


Figure 3.6 The plate in the developer.

Figure 3.7 The developed plate with the pattern visible.

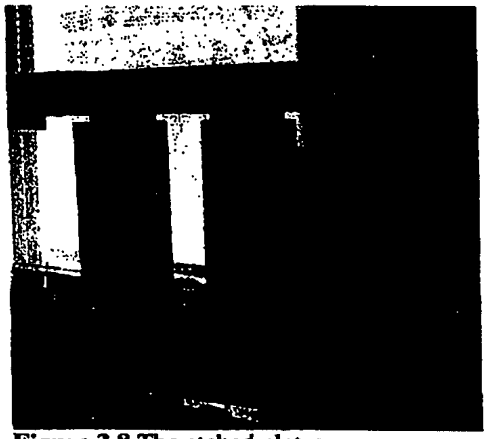


Figure 3.8 The etched plates.

Figure 3.9 The plate being wiped with acetone.

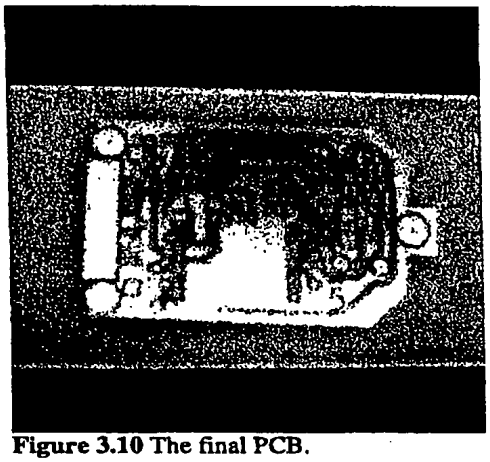


Figure 3.10 The final PCB.

The PCB was trimmed to its final shape by cutting of the superfluous plastic. To connect both sides of the plate, appropriate holes were drilled and filled with conductive material. The components were then soldered on their right positions and the PBA was placed in its aluminum/plastic box. Finally the microcontroller was flashed with the release software and the phone with its specific software.

3.2 The Software

The software consists of two different parts; the microcontroller software, for control of the entire accessory and the communication with the phone, and the phone software for sending data to the accessory.

3.2.1 The Microcontroller Software

The microcontroller software consists of three files; *TTS.c, int_routines.c* and *SPI_com.c*, see Appendix D. All code intended for the microcontroller is compiled by IAR Embedded Workbench, chapter 4.2, and executed in AVR Studio, chapter 4.3, with the emulator ICE200, chapter 4.4. After having a well-functioning code, the emulator is changed to a microcontroller which is flashed by AVR In-System Programmer (ISP), chapter 4.5.

3.2.1.1 TTS.c

In TTS.c both the microcontroller and the TTS circuit are initialized. The microcontroller's registers are set to ensure wanted functionality, see Appendix C. The TTS circuit is initialized by sending the right commands in the right order to it. These commands power up, start the crystal, set the speech pitch etc, see Appendix D.1.1.

The TTS.c is the main program and contains the following functions:

void init(void); initiates the microcontroller to wanted functionality by setting the ports to either inputs or outputs and by setting the registers.

void init_tts(void); initiates the TTS circuit by resetting it and sending start-up commands in a specific order according to the data sheets, see the code in Appendix D.

void reset(void); resets the TTS circuit (hardware reset). The TTS circuit has an internal power-on reset circuit that ensures correct initialization upon application of power. The reset pin signal must be held high for 0.5 µs to achieve a reset, see Figure 3.11.

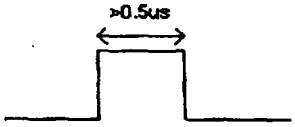


Figure 3.11 Timing for the reset condition.

void main(void); the main function with an infinite while-loop waiting for incoming data and button presses, see Figure 3.12.

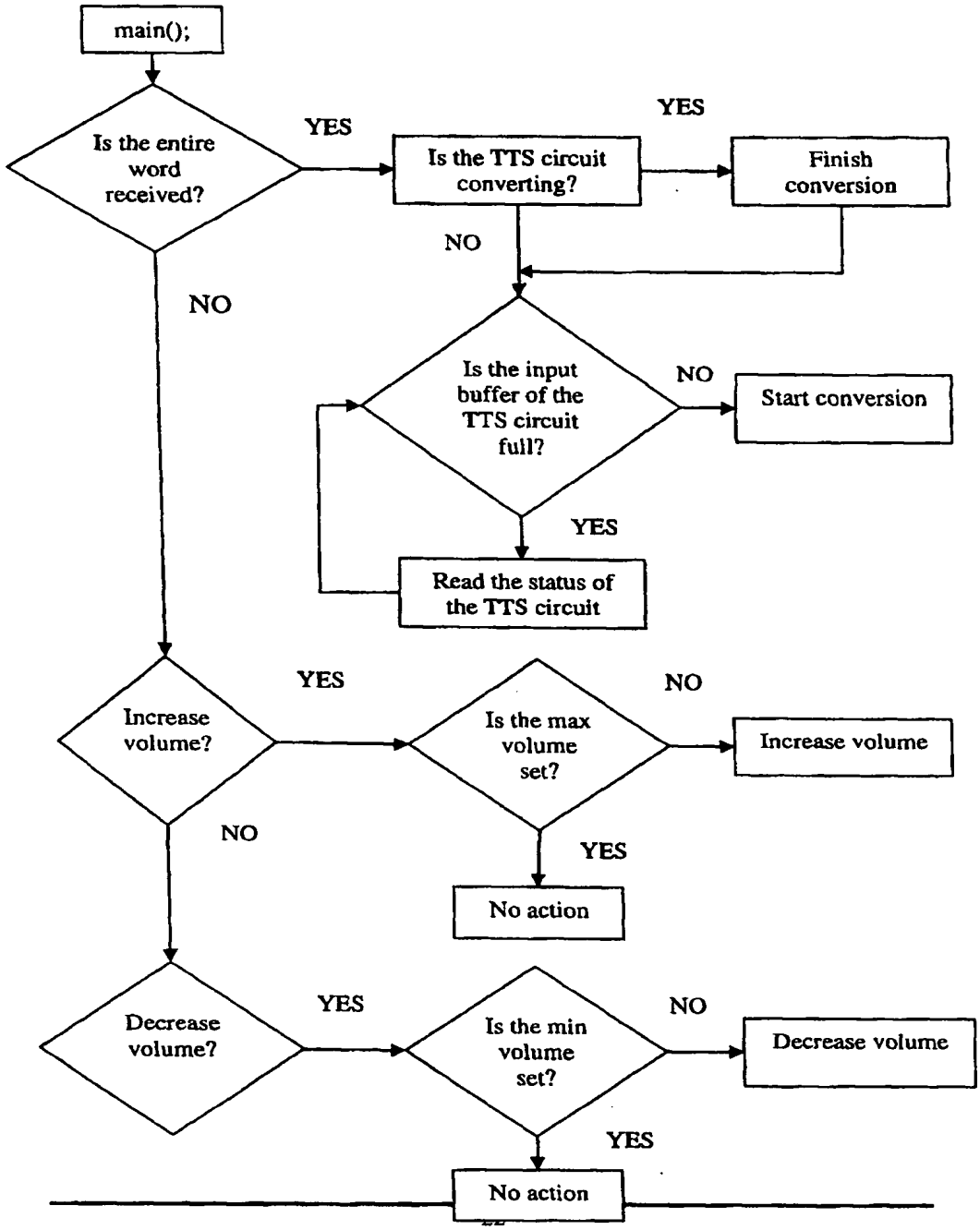


Figure 3.12 Flow chart over the *main()* function.

3.2.1.2 int_routines.c

In int_routines.c, see Appendix D.1.2, the external interrupts and data communication interrupts are handled. The microcontroller provides 12 different interrupt sources and each interrupt has a separate program vector in the program memory space. The interrupts have different priority levels and the lower the address the higher the priority. Each address source can be reached with a specific name indicating the interrupt referred to.

All the possible interrupts are not used in this application. The ones used are the following:

interrupt [INT0_vect] void INT0_interrupt(void); interrupt routine for the external interrupt that increases the volume, see Figure 3.13.

interrupt [INT1_vect] void INT1_interrupt(void); interrupt routine for the external interrupt that decreases the volume, see Figure 3.13.

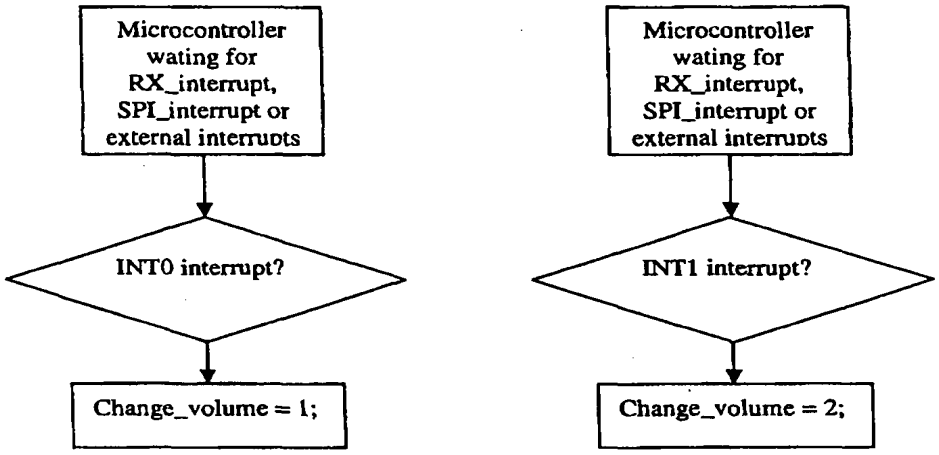


Figure 3.13 Flow chart over the external interrupt functions.

interrupt [SPI_STC_vect] void SPI_STC_interrupt(void); interrupt routine for reading status bytes from the TTS circuit, occurs when serial transfer is completed, Figure 3.14.

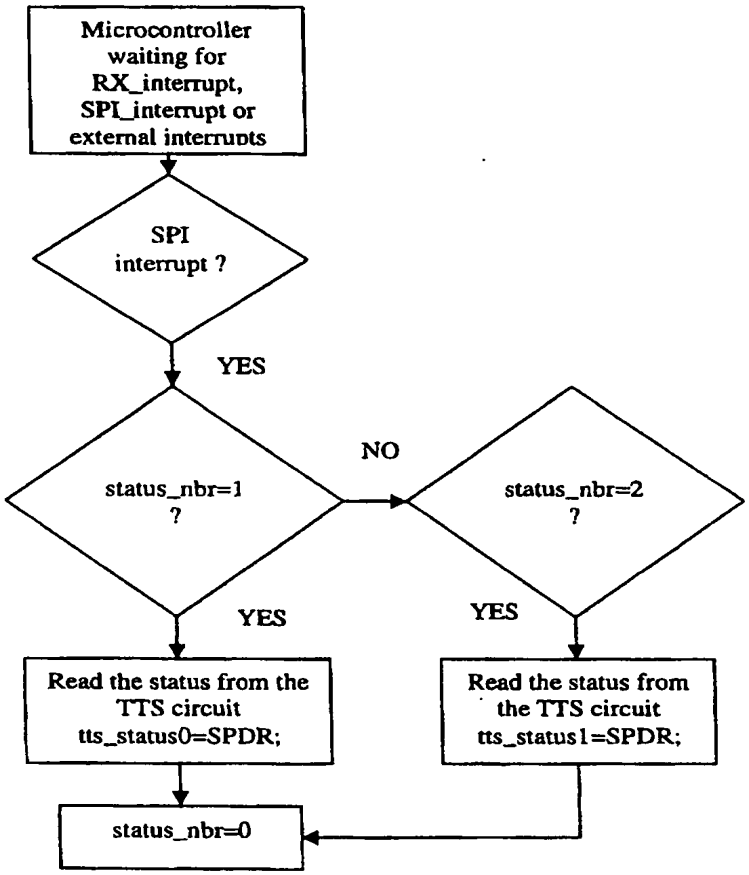
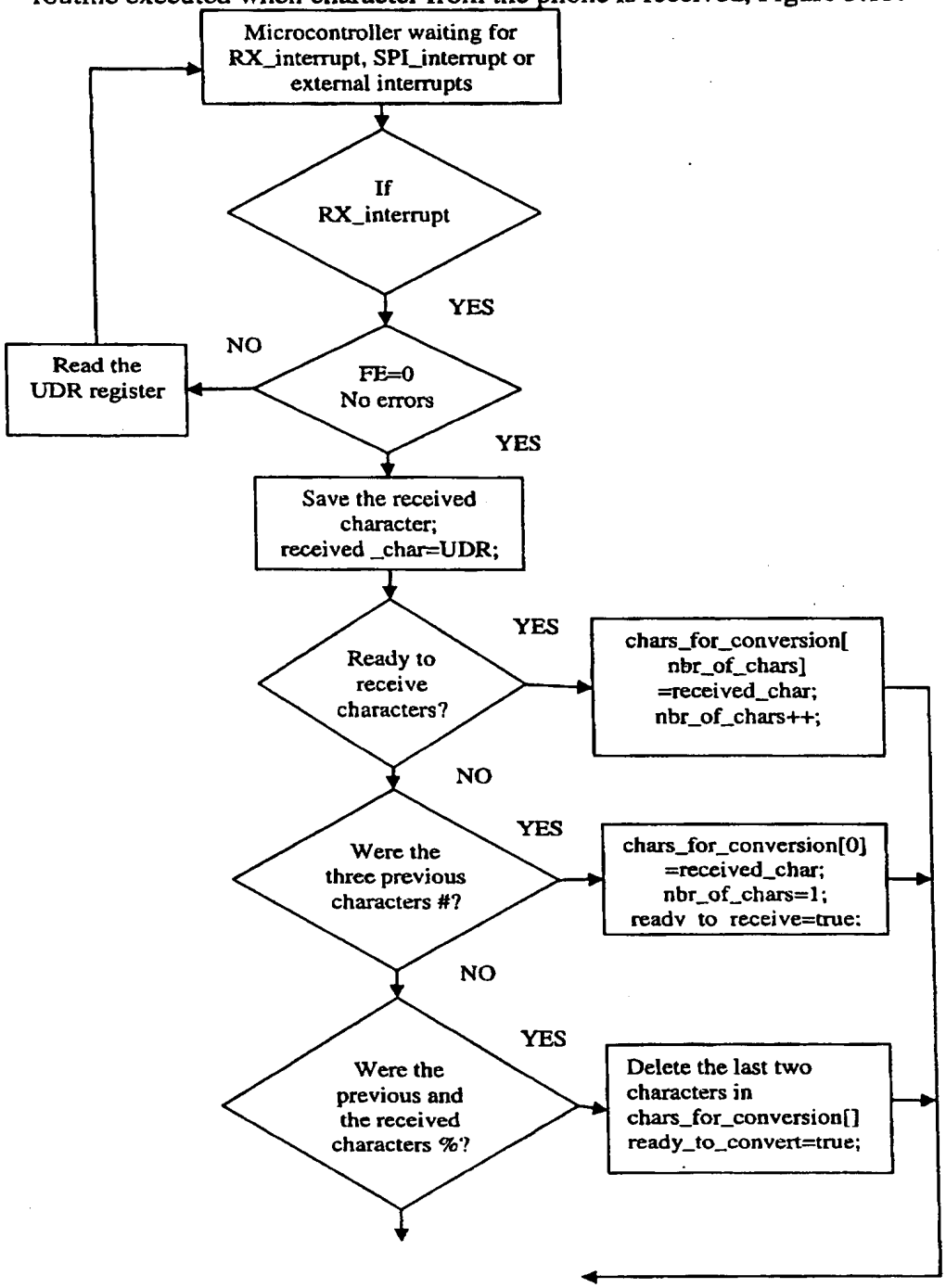


Figure 3.14 Flow chart over the SPI interrupt.

*interrupt [UART_RX_vect] **void** UART_RX_interrupt(void);* interrupt routine executed when character from the phone is received, Figure 3.15.



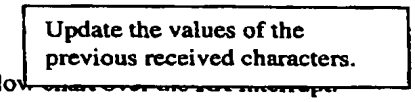


Figure 3.15 Flow

3.2.1.3 SPI_com.c

The SPI.c contains the needed functions for communication between the microcontroller and the TTS circuit, see Appendix D.1.3. To initiate a transfer the SS pin on the TTS circuit must go from high to low condition and to finish a transfer it must go high again.

The following functions are needed for SPI communication:

***void** send_cmd(**char** cmd_byte, **char** cmd_data_byte);* used for sending commands to the TTS circuit, see Figure 3.16.

***void** send_proc(**char** cmd_byte, **char** cmd_data_byte);* used for shifting command bytes to the TTS circuit, see Figure 3.16.

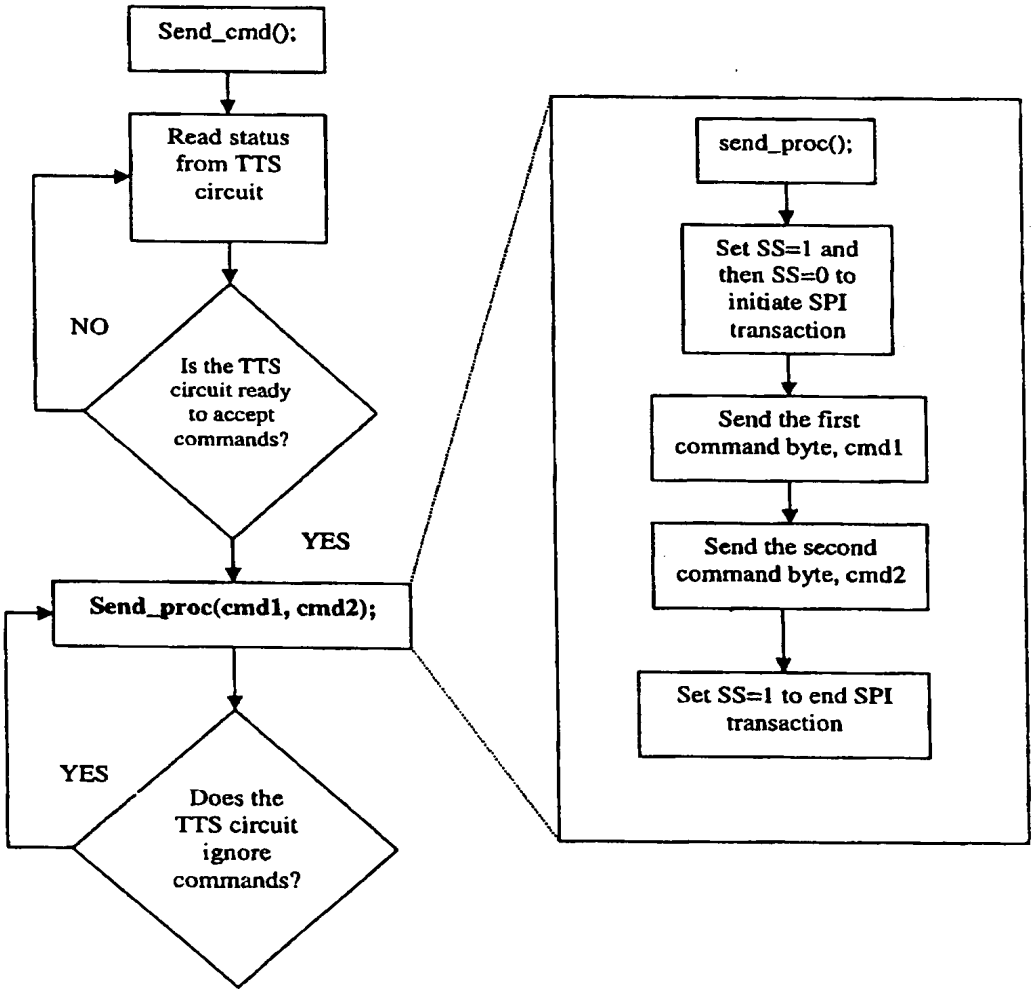


Figure 3.16 Flow chart over the functions *send_cmd()* and *send_proc()*.

void convert(void); used for sending data to the TTS circuit, Figure 3.17.

void send_data(void); used for shifting data bytes to the TTS circuit.

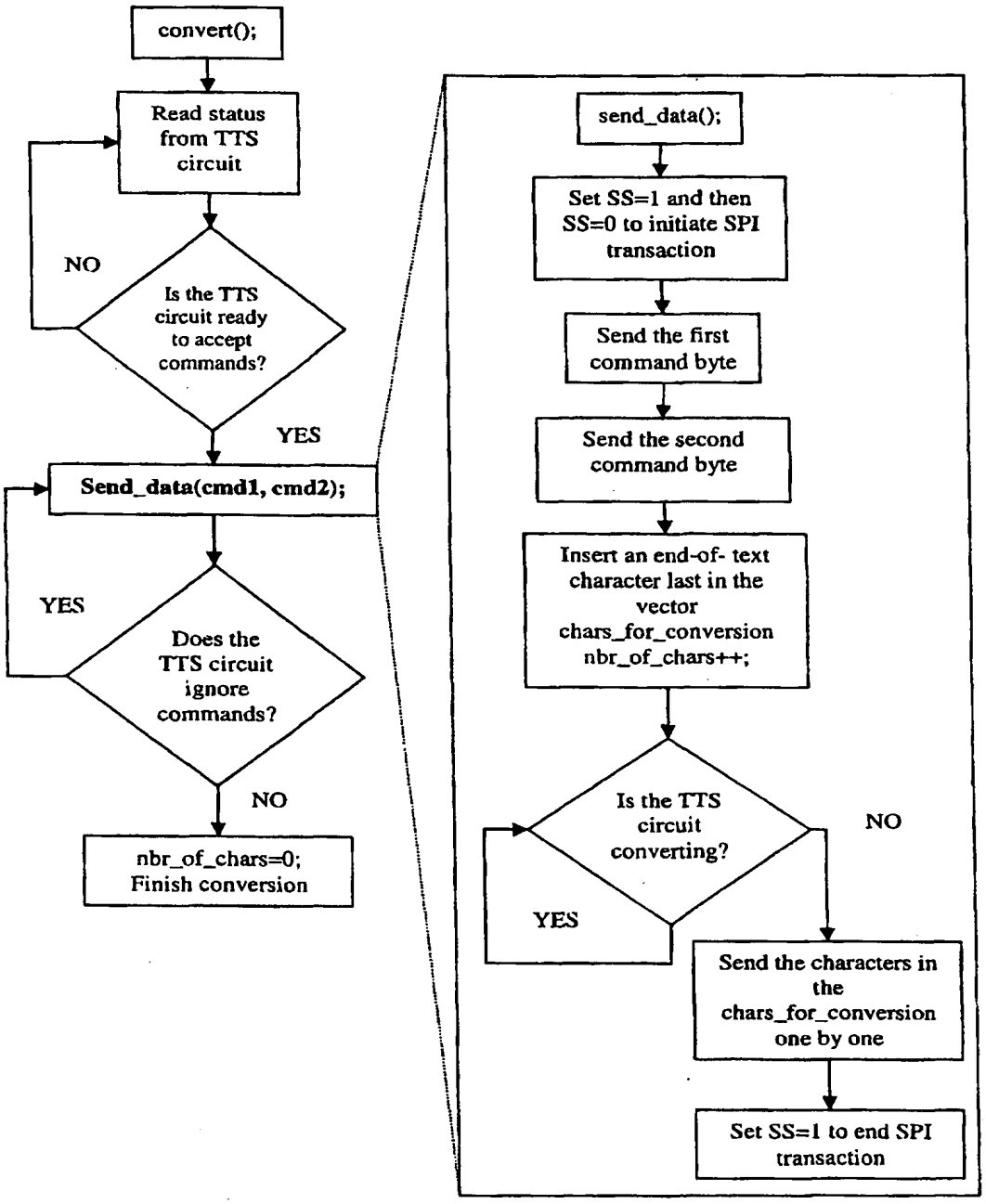


Figure 3.17 Flow chart over the functions *convert()* and *send_data()*.

The three files mentioned above and how their different functions interact with each other is shown in Figure 3.18. The function pointed at is the function called from the one pointing.

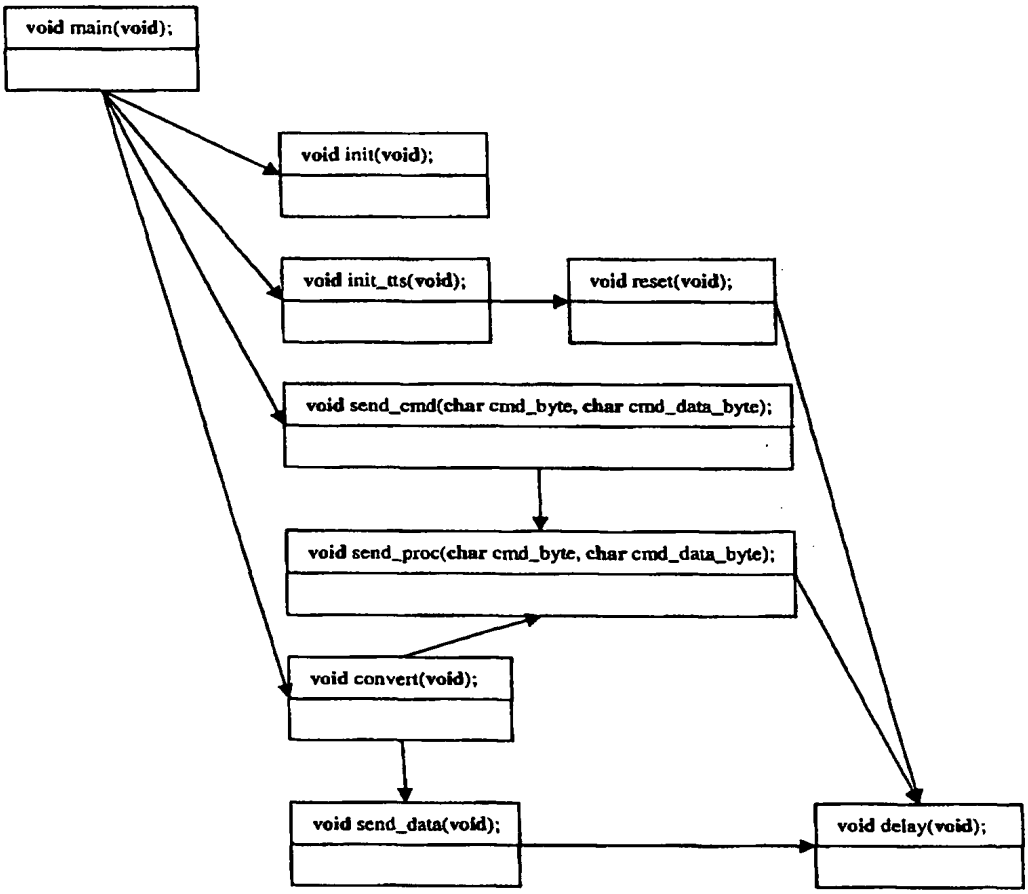


Figure 3.18 Block diagram over the interaction between the functions.

3.2.2 The Phone Software

All phone software is written in Visual C++ and is configuration managed by ClearCase and CME2, see chapter 4.6.

A similar code as below was added to the modules with differences like the names of the text ids and how to reach them.

```
length  = TextGetLength(item_p->textId);
buf  = ALLOC_OBJMMI_DATA(WCHAR_t, (length+1));
label = ALLOC_OBJMMI_DATA(unsigned char, (length+1));
bufSize  = TextGet((item_p->textId), buf, (length+1));
wstrntostr(label, buf, bufSize);
printf("\n%s%s%s\n", "###", label, "%%");
FREE_OBJMMI_DATA(&buf);
FREE_OBJMMI_DATA(&label);
```

The affected modules and files are:

- BA_SoftwareModule_SMS\cnh1010079_uimessaging\internsoftware\
  - sms_read.c: ReadMsg_Enter_Action(....){
- LD_SoftwareModules_002\cnh1010012_userint\internsoftware\disp\objects\navspace\
  - dsp_basl_laf.c: BaseListLaF_Impl_DrawItem(....){
  - dsp_desktop_laf.c: DesktopLaF_Impl_DrawItem(....){
- LD_SoftwareModules_002\cnh1010012_userint\internsoftware\disp\objects\
  - dsp_1ofm.c: OneOfManySelectionShowLabel(....){
  - dsp_nofm.c: NoOfManySelectionShowLabel(....){
  - dsp_onof.c: OnOffInputDraw(....){
  - dsp_feedback.c: FeedbackDraw(....){

To be able to call the function *wstrntostr(label, buf, bufSize);* the file *error_r_txt_ui.h* has to be included in the files above.

The phone transmits data through its system contact with a default baud rate of 115200 bps. The microcontroller in the accessory achieves the lowest possible baud rate error at 9600 bps and therefore the baud rate is set to this level both in the microcontroller and in the phone. The phone baud rate can be changed in the file *\hardware.h*:

```
#define UART_MAIN_CTRL_BAUD_RATE_115200          (0x00)
#define UART_MAIN_CTRL_BAUD_RATE_9600            (0x00)
```

The hex value for *UART_MAIN_CTRL_BAUD_RATE_115200* is default 0xXX and has to be changed to 0x00. This file contains static definitions of macros. When changing a replacement value (the hex value) in this file it will affect all files that use the particular macro.

When the phone is logged it constantly transmits data, such as radio parameters, on its system bus. Since the microcontroller handles all data that it receives on its UART it can easily become overloaded. This is solved by disabling the unnecessary prints, all of them except for the ones

at the phone start up process. This is done in the file in *\DescrExtra.cfg* by writing:

[SourceFiles_Options]
mph + ALL=(-UPRINT_A_)*
rr.c + ALL=(-UPRINT_A_)
msgmngr.c + ALL=(-UPRINT_A_)

This file can be used to apply private temporary additions to a product i.e. it can typically be used to override files and options in other files.

4 The Tools

The tool used for drawing the circuit diagram and the placement diagram in this project was Easy-PC.

Developing the microcontroller software required several tools. IAR Embedded Workbench was used for compiling the software and AVR Studio version 4.0 was used for executing it. These were used together with the emulator ATICE200 version 1.10 and finally AVRISP was used for programming the microcontroller. During programming, the TTS circuit must be disconnected since it also uses the microcontroller's SPI interface. This is achieved by switching the coupling manually, see Appendix A.

ClearCase and CME2, see chapter 4.6, are used by Sony Ericsson Mobile Communications for handling the different versions of the phone software and therefore also used in this project. The tool for developing the phone software was Visual C++.

4.1 Easy-PC Number One System

Easy-PC Number One System is an electrical CAD program used for creating circuit diagrams, placement diagrams and templates for Printed Circuit Boards, PCBs. The program contains a number of components in its default library, but the user has the opportunity to create new ones when needed. To do this, first the schematic symbol must be created. The symbol does not have to look exactly as the specific circuit but it is important that it has the right number of pins. To every schematic symbol a corresponding PCB symbol must be created. The PCB symbol must have the exact dimensions as the actual circuit, concerning both shape and pin layout. When both the schematic and the PCB symbol have been created the pins must be matched, meaning that the symbolic pin numbers are paired with the corresponding numbers on the PCB symbol.

When all the needed symbols are available the circuit diagram can be drawn, see Appendix A. When this is done the program can easily (by a button press) translate it to the template for the PCB. The placement drawing, see Appendix B, can then be created by manually placing the components to the wanted positions on the PCB.

4.2 IAR Embedded Workbench

IAR Embedded Workbench provides a powerful environment for developing projects with a range of different target processors. It has been specially designed to fit in with the way software development projects are

typically organized. It allows projects to be organized in a hierarchical tree structure showing the dependency between files at a glance.

4.3 Atmel AVR Studio

AVR Studio is used for execution of program code. It is an integrated development environment for writing and debugging AVR applications. It combines an editor, a project manager, an assembler/compiler interface and a debugger in one development tool. The user can manage projects, edit source code, simulate or interface an emulator.

AVR Studio can be used together with IAR in such way that after writing and compiling the code in IAR, the compiled code can be opened in AVR Studio. If an emulator is detected on the serial port, AVR Studio automatically enters emulation mode, otherwise it starts in simulation mode. The program detects the target system and downloads the code to it. When the program is downloaded the user can set an unlimited number of breakpoints and can also watch the registers, SRAM, EEPROM, Flash and I/O as they change their values.

4.4 Atmel ATICE200

ATICE200 is a real-time in-circuit emulator. This is a kit that contains several personality adapters with different numbers of pins for use in different purposes. Each adapter corresponds to one or two of Atmels processors in the AT90S-series. The adapter used in this project is the Atadap3000 that corresponds to AT90S8515. The ICE200 provides a debugging platform with a minimum of differences between the emulator and the actual processor it is emulating. This provides identical electrical characteristics. Figure 4.1 shows the emulator with its adapter connected to the wired prototype.

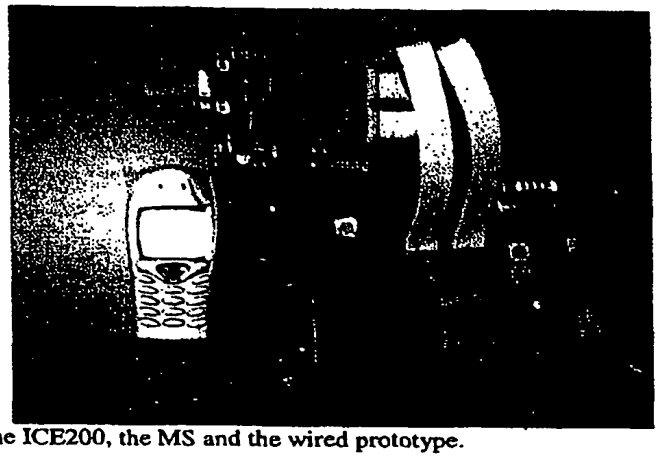


Figure 4.1 The ICE200, the MS and the wired prototype.

When used with the AVR Studio debugging environment, the ICE200 gives the user full run-time control, unlimited number of breakpoints, symbolic debugging and full memory and register visibility.

The main board contains the program memory that holds the application code that is being emulated. It also contains logic for communicating with the host PC, and the breakpoint logic. The pod contains the AVR emulator chip and the personality adapters map the pin out from ICE200 pod to each microcontroller it supports. The adapter includes an identification code that AVR Studio uses for automatic device type detection. These parts are connected to the target application.

4.5 Atmel AVRISP

AVR In-System Programmer, AVRISP, is used for flashing AVR microcontrollers. The programmer connects to a PC through a standard RS232 serial interface and uses AVR Studio as front-end software. The programmer uses the SPI interface to flash the microcontroller.

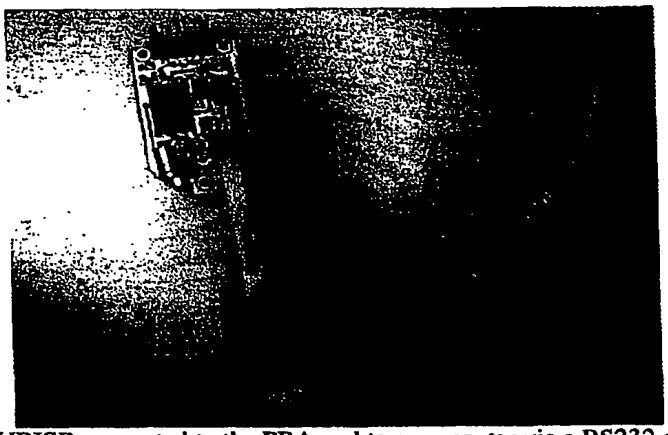


Figure 4.2 AVRISP connected to the PBA and to a computer via a RS232 cable.

4.6 ClearCase and CME2

ClearCase and CME2 are two programs used for software configuration management that helps automate the tasks required to write, release, and maintain software. They track changes to every file and directory in the projects and support parallel development. CME2 is developed by Ericsson while ClearCase is a commercial program that is publicly available.

5 Result

The purpose of this thesis was to develop a demonstrator that could read SMS and menus. The result was a well functioning prototype that fulfils the specified behavior. Both the hardware and the software works fine with just the few limitations discussed in chapter 5.2. Since this application is useful not only for the visually impaired but also for example as a complement for vehicle hands free, future development is not excluded.

5.1 How to use the accessory

To be able to use the accessory, a Sony Ericsson T68i with special software is required.

- Make sure that the battery is charged and that the phone is turned off.
- Attach the accessory to the phone.
- Turn on the phone; the accessory is ready for use.
- To change volume, pull the button on the accessory upwards or downwards. There are eight volume levels and level four is set as default.
- To use headphones insert the connector in the tele socket.
- Before removing the accessory, turn off the phone.

5.2 Limitations

There are a few limitations that have to be considered when using the accessory:

- The accessory can not be attached to any phone. The phone must have the special software mentioned above.
- The accessory must be attached to the phone before the phone is turned on.
- Since it was not possible to communicate with AT commands only the menus in the modified modules are read.
- When browsing fast, the TTS circuit does not manage to handle all data sent to it since it turns off reception while converting, and therefore all menus passed will not be read.
- If removed when the phone is on, the phone will hang.
- The power consumption is rather high and a battery will not last for long.

5.3 Problems

During the development mainly three problems appeared; the SPI communication, the loss of AT commands and settings of the linker in IAR Embedded Workbench.

The problem with the SPI communication was the clock synchronization between the microcontroller and the TTS circuit. It is important to have the same clock phase and the same clock polarity i.e. the microcontroller must be set to have the same polarity and phase as the circuit it communicates with, in this case the TTS circuit. To find out which these settings are the data sheets must be carefully studied and measurements must be done.

The loss of AT commands was a major drawback which meant more work since the software modifications had to be made in more than one module and file. If they existed, the phone software would probably not have been affected at all, and everything could be handled and controlled by the microcontroller.

The final problem rose when the microcontroller was to be flashed. The linker must create a release file of the right format for the specific microcontroller to be used. This requires that the right xcl-file, lnk8515_512.xcl as in this case, is selected in the settings of the IAR Embedded Workbench, see Appendix E.1. If this file does not exist, as the problem was in this case, it can be downloaded from Atmels homepage. Usually the files are named after the corresponding microcontroller. Even if the wrong file is selected the AVRISP indicates that the flashing succeeded, but the microcontroller will not behave as expected.

6 Discussion

For future development it is not necessary to restrict to a hardware solution in form of an accessory as in this case. It is fully possible to integrate the corresponding solution in the phone, either as an ASIC or as a pure software solution, considering that the phones today already contain everything needed. Since the interest for TTS is large in many places, this feels to be right in time.

To offer TTS in the phone is not only an aid for the visually impaired and the drivers but also a step further in personalizing the phone. Today personalization can be done by changing the panels, having different polyphonic sounds, the possibility to add pictures etc. Just imagine how it would be to have your favorite artist reading your SMS, by combining TTS with MIDI files. This brings whole new possibilities to multimedia, for example games.

The TTS can also be considered in combination with WAP and the Yellow Pages route service or with a future GPS function in the phone. Together with memory sticks it can be possible to have your favorite books read while sitting in the car, on the train or just relaxing in the hammock. This is only a small fraction of all possible features and imagination is the only limit.

References

[1] Ericsson, *System Bus Specification for the Digital System Phones, CT/UT 93:003*, 1995-06-14

[2] IAR Systems, EWA90-1, *AT90S Embedded Workbench Interface Guide for Atmel's AT90S Microcontroller Family*, September 1996

[3] IAR Systems, ICCA90-1, *AT90S C Compiler Programming Guide for Atmel's AT90S Microcontroller Family*, September 1996

[4] Atmel, *AVR ICE200 User Guide*, 1999, www.atmel.com

[5] Ericsson, *Description of System Connector Electrical Interfaces for the Marianne Product Platform, TX/B 97:0028*, 2000-08-10

[6] Number One Systems, *Easy-PC for Windows*, September 2000

[7] National Semiconductor, *LM4894 Boomer*, June 2001, www.national.com

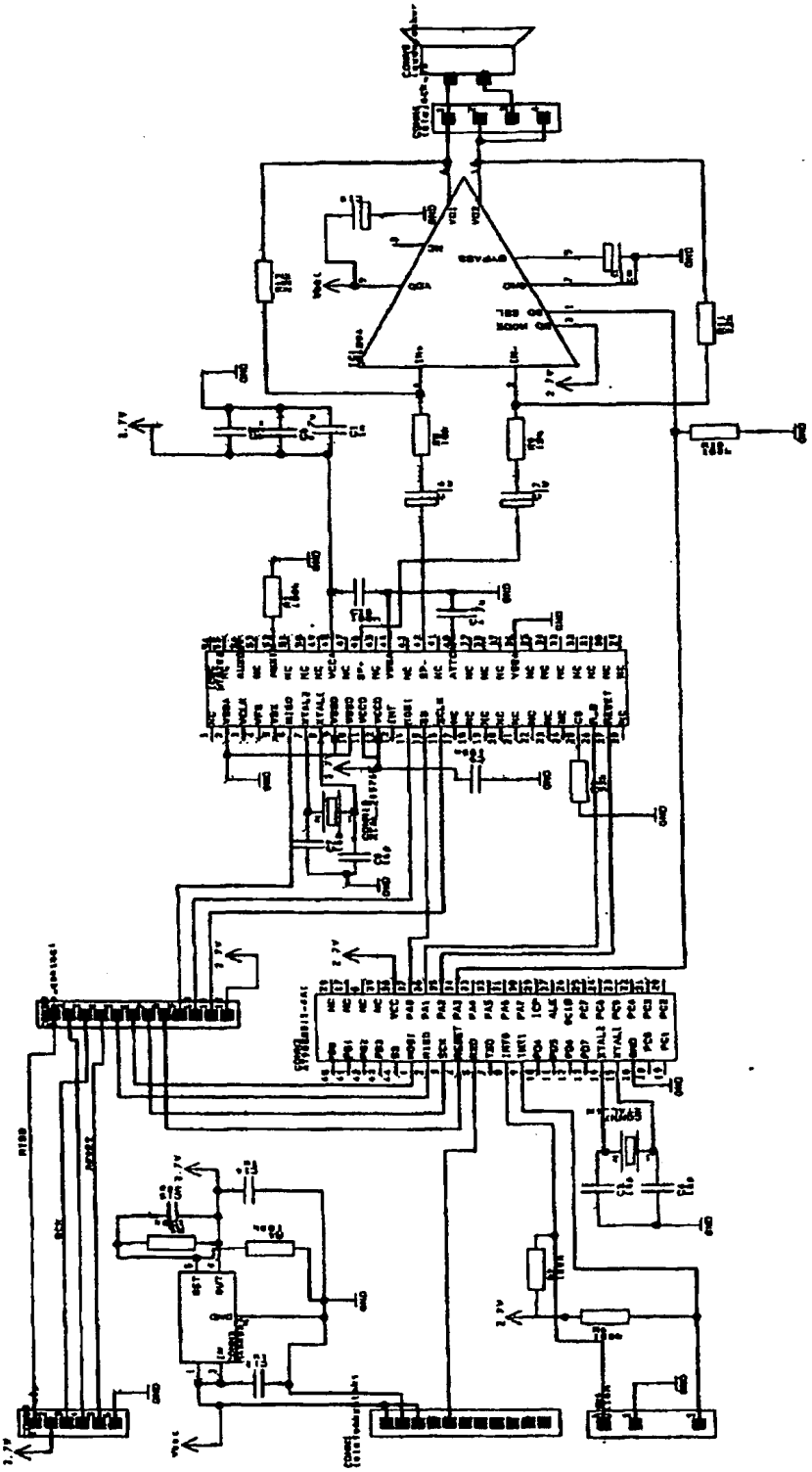
[8] Atmel, *8-bit AVR Microcontroller with 8K Bytes In-System Programmable Flash AT90S8515*, Rev 0841G-09/01, www.atmel.com

[9] Saar Hezi, *Designing in Text-to-Speech Capability for Portable Devices*, Winbond, April 2002

[10] Winbond, *WTS701 Winbond Single-Chip Text-To-Speech Processor*, Rev 3.07, www.winbond-usa.com

[11] Maxim, *MAX8863*, www.elfa.se

Appendix A – Circuit Diagram



Appendix B – PCB and PBA

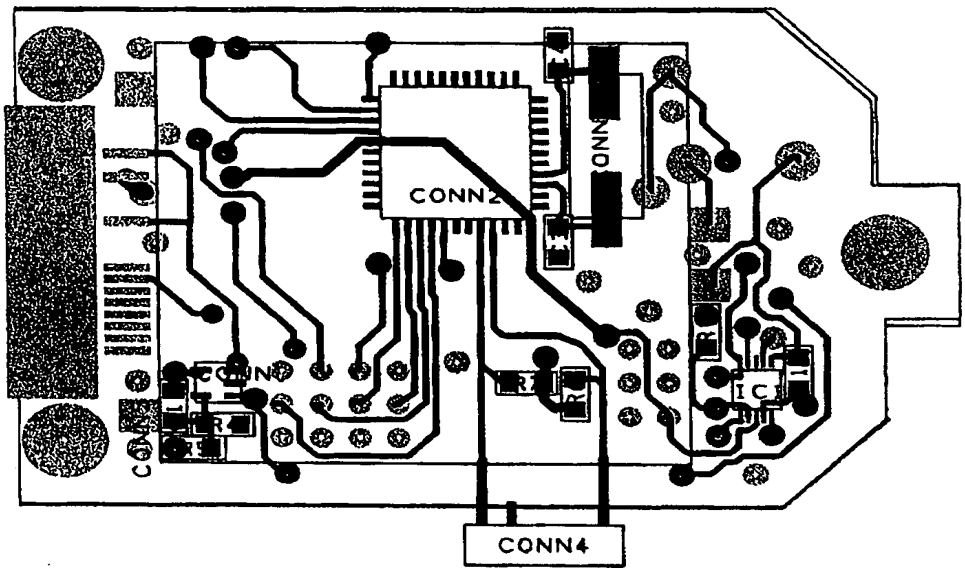


Figure B.1 The front side of the PCB.

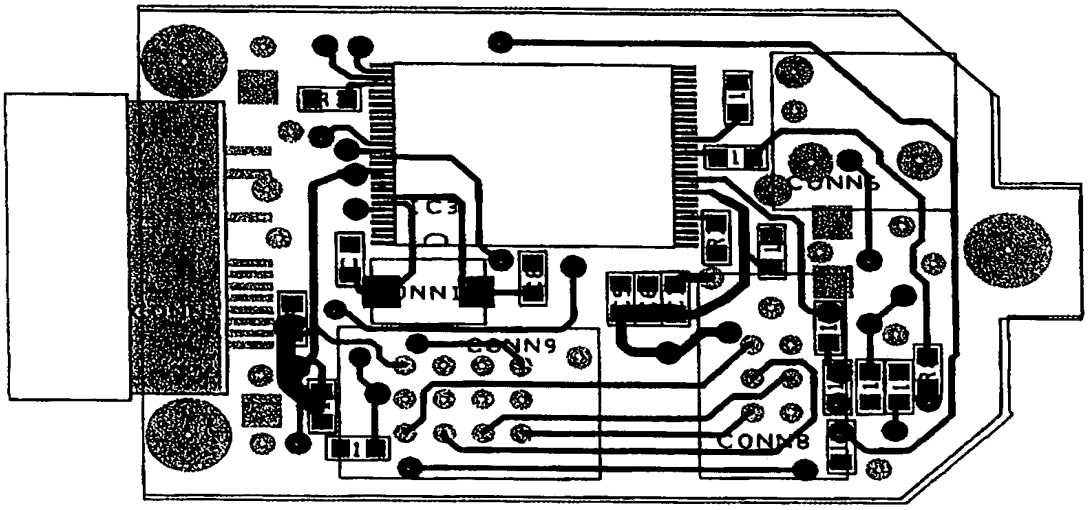


Figure B.2 The back side of the PCB.

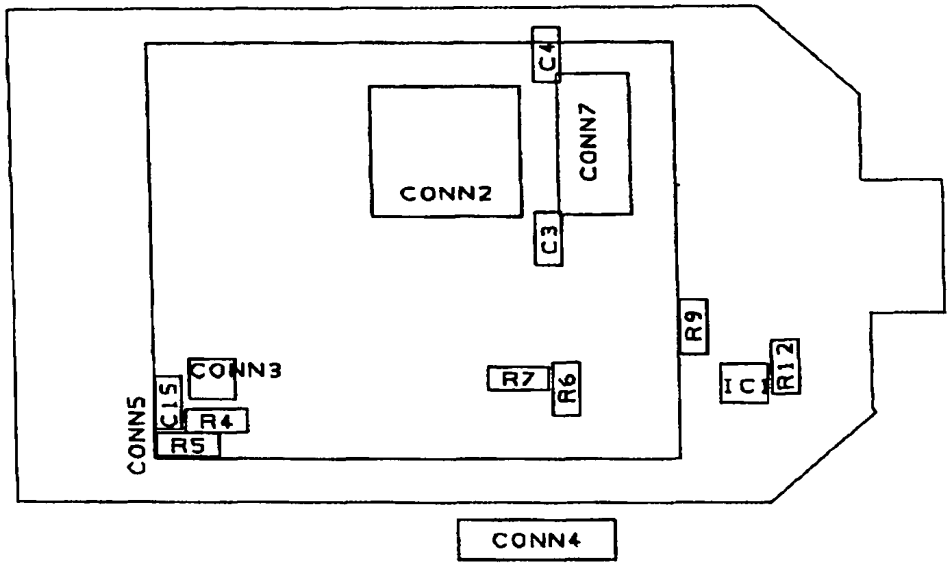


Figure B.3 The front side of the PBA.

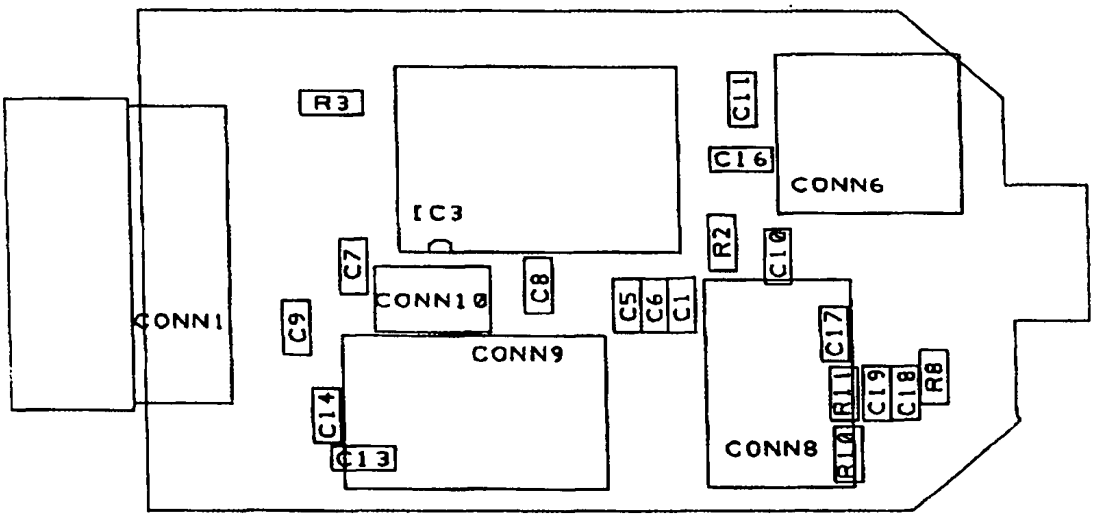


Figure B.4 The back side of the PCB.

List of components

CONN1	System connector
CONN2	AT90S8515
CONN3	MAX8863
CONN4	Button
CONN5	Loudspeaker
CONN6	Tele socket
CONN7	Crystal – 4MHz
CONN8	Flash socket for AVRISP
CONN9	Socket for manual reconnection of SPI
CONN10	Crystal – 24.576MHz
IC1	LM4894MM
IC3	WTS701
R2	100kΩ
R3	33 kΩ
R4	100 kΩ
R5	120 kΩ
R6	100 kΩ
R7	100 kΩ
R8	10 kΩ
R9	10 kΩ
R10	100 kΩ
R11	22 kΩ
R12	22 kΩ
C1	1nF
C3	15pF
C4	15pF
C5	82nF
C6	4.7µF
C7	15pF
C8	15pF
C9	100nF
C10	100nF
C11	4.7 µF
C13	1 µF
C14	1 µF
C15	20pF
C16	1 µF
C17	1 µF
C18	1 µF

C19	1 µF

Appendix C – Register settings of AT90S8515

SREG – Status Register

I	T	H	S	V	N	Z	C
1	0	0	0	0	0	0	0

I – Global Interrupt Enable: Is set to enable global interrupts.
T – Bit Copy Storage: Not used in this application. Initial value is set to zero.
H – Half-carry Flag: Not used in this application. Initial value is set to zero.
S – Sign Bit: Not used in this application. Initial value is set to zero.
V – Two's Complement Overflow Flag: Not used in this application. Initial value is set to zero.
N – Negative Flag: Not used in this application. Initial value is set to zero.
Z – Zero Flag: Not used in this application. Initial value is set to zero.
C – Carry Flag: Not used in this application. Initial value is set to zero.

SP – Stack Pointer

Not used in this application.

GIMSK – General Interrupt Mask Register

INT1	INT0	-	-	-	-	-	-
1	1	-	-	-	-	-	-

INT1 – External Interrupt Request 1 Enable: Set together with SREG:I enables external interrupts. Decrease the volume.
INT0 – External Interrupt Request 0 Enable: Set together with SREG:I enables external interrupts. Increase the volume.
The rest of the bits are reserved.

GIFR – General Interrupt Flag Register

INTF1	INTF0	-	-	-	-	-	-
1	1	-	-	-	-	-	-

INTF1 – External Interrupt Flag 1: Is set to clear interrupt flag.
INTF0 – External Interrupt Flag 1: Is set to clear interrupt flag.
The rest of the bits are reserved.

TIMSK – Timer/Counter Interrupt Mask Register

TOIE1	OCIE1A	OCIE1B	-	TICIE1	-	TOIE0	-
0	0	0	-	0	-	0	-

TOIE1 – Timer/Counter1 Overflow Interrupt Enable: Not used in this application. Initial value is set to zero.

OCIE1A – Timer/Counter1 Output CompareA Match Interrupt Enable: Not used in this application. Initial value is set to zero.
OCIE1B – Timer/Counter1 Output CompareB Match Interrupt Enable: Not used in this application. Initial value is set to zero.
TICIE1 – Timer/Counter1 Input Capture Interrupt Enable: Not used in this application. Initial value is set to zero.
TOIE0 – Timer/Counter0 Overflow Interrupt Enable: Not used in this application. Initial value is set to zero.
The rest of the bits are reserved.

TIFR – Timer/Counter Interrupt Flag Register

Not used in this application.

MCUCR – MCU Control Register

SRE	SRW	SE	SM	ISC11	ICS10	ISC01	ISC00
0	0	0	0	1	1	1	1

SRE – External SRAM Enable: Not used in this application. Initial value is set to zero.
SRW – External SRAM Wait State: Not used in this application. Initial value is set to zero.
SE – Sleep Enable: If set SLEEP is enabled. Is set just before execution.
SM – Sleep Mode: Is cleared to select Idle Mode.
ISC11 – Interrupt Sense Control 1, Bit 1: Is set to activate interrupt 1 at rising edge.
ISC10 – Interrupt Sense Control 1, Bit 0: Is set to activate interrupt 1 at rising edge.
ISC01 – Interrupt Sense Control 0, Bit 1: Is set to activate interrupt 0 at rising edge.
ISC00 – Interrupt Sense Control 0, Bit 0: Is set to activate interrupt 0 at rising edge.

TCCR0 – Timer/Counter0 Control Register

Not used in this application.

TCNT0 – Timer Counter0

Not used in this application.

TCCR1A – Timer/Counter1 Control Register A

Not used in this application.

TCCR1B – Timer/Counter1 Control Register B

Not used in this application.

TCNT1H and TCNT1L – Timer/Counter1

Not used in this application.

***OCR1AH and OCR1AL** – Timer/Counter1 Output Compare Register*
Not used in this application.

***OCR1BH and OCR1BL** – Timer/Counter1 Output Compare Register*
Not used in this application.

***ICR1H and ICR1L** – Timer/Counter1 Input Capture Register*
Not used in this application.

***WDTCR** – Watchdog Timer Control Register*
Not used in this application.

***EEARH and EEARL** – EEPROM Address Register*
Not used in this application.

***EEDR** – EEPROM Data Register*
Not used in this application.

***EECR** – EEPROM Control Register*
Not used in this application.

***SPCR** – SPI Control Register*

SPIE	SPE	DORD	MSTR	CPOL	CPHA	SPR1	SPR0
1	1	0	1	1	1	0	0

SPIE – SPI Interrupt Enable: Is set to enable SPI interrupts.
SPE – SPI Enable: Is set to enable SPI.
DORD – Data Order: The MSB of the data word is transmitted first when cleared.
MSTR – Master/Slave Select: Is set to select Master SPI Mode.
CPOL – Clock Polarity: Is set to make SCK high when idle.
CPHA – Clock Phase: Selectable.
SPR1 – SPI Clock Rate Select 1: Control the SCK rate of the master.
SPR0 – SPI Clock Rate Select 0: Control the SCK rate of the master.

***SPSR** – SPI Status Register*
Read only.

***SPDR** – SPI Data Register*
Writing to the register initiates a data transmission and will be set just before transmission.

***UDR** – UART I/O Data Register*
It alternates between being a register storing incoming data and outgoing data.

USR – UART Status Register

RXC	TXC	UDRE	FE	OR	-	-	-
-	1	-	-	-	-	-	-

RXC – UART Receive Complete: Is set when received character is transferred from Receiver Shift register to UDR.
TXC – UART Transmit Complete: Is set when the entire character is shifted out from Transmit Shift register and no new data is written to UDR. Not used.
UDRE – UART Data Register Empty: Zero indicates that the transmitter is ready to receive a new character for transmission.
FE – Framing Error: Is set if a framing error is detected.
OR – Overrun: Is set if an overrun condition is detected.

UCR – UART Control Register

RXCIE	TXCIE	UDRIE	RXEN	TXEN	CHR9	RXB8	TXB8
1	0	0	1	0	0	-	-

RXCIE – RX Complete Interrupt Enable: Is set to enable the Receive Complete Interrupt routine.
TXCIE – TX Complete Interrupt Enable: Transmit Complete Interrupt routine is not used.
UDRIE – UART Data Register Empty Interrupt Enable: Enables the UART Data Register Empty Interrupt routine if set.
RXEN – Receiver Enable: Is set to enable the UART receiver.
TXEN – Transmitter Enable: Is not used.
CHR9 – 9-bit Characters: Makes characters 9 bits long plus start and stop bits if set.
RXB8 and TXB8 depends on CHR9, contain the ninth bit.

UBRR – UART BAUD Rate Register

MSB							LSB
0	0	0	1	1	0	0	1

UBRR set to 25 corresponds to a BAUD rate of 9600 when a 4 MHz crystal is used.

ACSR – Analogue Comparator Control and Status Register

ACD	-	ACO	ACI	ACIE	ACIC	ACIS1	ACIS0
1	-	-	0	0	0	0	0

ACD – Analogue Comparator Disable: Is set to switch of the power to the Analogue Comparator.
The other bits are not used in this application.

DDRA – Port A Data Direction Register

DDA7	DDA6	DDA5	DDA4	DDA3	DDA2	DDA1	DDA0
1	1	1	1	1	1	0	1

The Port A pins are set as outputs, except for pin A1.

DDRB – Port B Data Direction Register

SCK	MISO	MOSI	SS	AIN1	AIN0	T1	T0
1	0	1	1	1	1	1	1

SCK – Serial Clock: Master clock output.
MISO – Master In Slave Out: Master data input.
MOSI – Master Out Slave In: Master data output.
SS – Slave Select: Master select output.
AIN1 – Analogue Comparator Negative Input: Not used in this application.
AIN0 – Analogue Comparator Positive Input: Not used in this application.
T1 – Timer/Counter1 Counter Source: Not used in this application.
T0 – Timer/Counter0 Counter Source: Not used in this application.

DDRC – Port C Data Direction Register

DDC7	DDC6	DDC5	DDC4	DDC3	DDC2	DDC1	DDC0
1	1	1	1	1	1	1	1

The Port C pins are set as outputs.

DDRD – Port D Data Direction Register

DDD7	DDD6	OC1A	DDD4	INT1	INT0	TXD	RXD
1	1	1	1	1	1	1	0

OC1A – Output Compare Match Output: Not used in this application.
INT1 – External Interrupt Source 1: Volume button.
INT0 – External Interrupt Source 0: Volume button.
TXD – Transmit Data: Is set to output.
RXD – Receive Data: Is set to input.
Other pins are not used in this application.

Appendix D – The Software

D.1 The C-files

D.1.1 TTS.c

```
/*=====================================*/
/*  Filename: TTS.c                    */
/*  Date: 02-12-20                     */
/*  Written by: Nercivan Kerimovska    */
/*          Anna Tomasson              */
/*=====================================*/

#include <io8515.h>
#include <ina90.h>
#include <int_routines.h>
#include <SPI_com.h>

/* Initiates processor   */
void init(void);

/* Initiates the text-to-speech circuit   */
void init_tts(void);

/* Resets the Text-to-speech circuit, HW reset  */
void reset(void);

void main(void)
{
 init();
 init_tts();

 while(1){
  if (ready_to_convert == 1){
   if ((tts_status0 & 0x02) == 0x02){          //Check if the tts is converting
    send_cmd(0x4C, 0x00);                      //FIN = finish
    send_cmd(0x06, 0x00);                      //RINT = read interrupt
   }
   while ((tts_status0 & 0x04) != 0x04){       //Check if the input buffer of the TTS
                                               circuit is full
    send_cmd(0x04, 0x00);                      //RDST = read status
   }
   convert();
   ready_to_convert = 0;
  }

  if (change_volume == 1){                     //Increase volume
   if (volume > 0){
    volume --;
    send_cmd(0x51, (0x30 + volume));           //SVOL = set the volume level to volume
```

```
    }
    change_volume = 0;
   }
   else if (change_volume == 2){                    //Decrease volume
    volume ++;
    if (volume > 7){
     volume = 7;
    }
    else{
     send_cmd(0x51, (0x30 + volume));               //SVOL = set the volume level to volume
    }
    change_volume = 0;
   }
  }
}

/* Processor initialization  */
void init(void)
{
 /* Sets the ports to outputs(=1) or inputs(=0)  */
 DDRA = 0xFD;                                       //All pins set as outputs except for pin A1
 DDRB = 0xBF;                                       //SPI interface
 DDRC = 0xFF;                                       //All pins set as outputs
 DDRD = 0xF3;                                       //All pins set as outputs except for INT0
                                                    and INT1

 /* Enables the needed interrupts  */
 SREG = 0x80;                                       //Enable global interrupts
 GIMSK = 0xC0;                                      //External interrupt enable
 GIFR = 0xC0;                                       //External interrupt flag, clear interrupt
                                                    flag
 MCUCR = 0x0F;                                      //Interrupt sense control, rising edge
                                                    (INT0 and INT1)

 /* Timer Counter Interrupt Mask Register */
 TIMSK = 0x00;                                      //Timer counter is disabled

 /*  SPI Control Register */
 SPCR = 0xDC;                                       //SPI enable, master select, SPI interrupts
                                                    disabled, CPOL = 1, CPHA = 1, SPR1 =
                                                    0, SPR0 = 0

 /* UART Control  */
 UCR = 0x90;                                        //UART transmit/receive enable and
                                                    interrupt enable
 USR = 0x40;                                        //UART transmit complete
 UBRR = 0x19;                                       //Baud rate (=9600)

 /* Analogue Comparator Control and Status Register  */
 ACSR = 0x80;                                       //Analogue comparator disabled

 PORTA &= 0xF7;                                     //Turns the amplifier on
}

/* Text-to-speech initialilzation  */
void init_tts(void)
```

```
{
 reset();
 send_cmd(0x10, 0x00);                  //RST = reset
 send_cmd(0x14, 0x00);                  //SCLC = set clock configuration
 send_cmd(0x02, 0x00);                  //PWUP = power up
 send_cmd(0x4E, 0x00);                  //SCOM = set com register
 //send_cmd(0x4F, 0x01);                //SCOD = set up codec configuration
 send_cmd(0x50, 0x43);                  //SAUD = set up audio control register
 send_cmd(0x51, 0x04);                  //SVOL = set the initail volume level
 send_cmd(0x52, 0x02);                  //SSPD = set SPD register
 send_cmd(0x77, 0x05);                  //SPTC = set speech pitch
 send_cmd(0x57, 0x00);                  //IDLE = idle
}

/* Resets the Text-to-speech circuit  */
void reset(void)
{
 PORTA &= 0xFB;
 PORTA |= 0x04;
 delay();
 delay();
 delay();
 PORTA &= 0xFB;
}
```

D.1.2 int_routines.c

```
/*=====================================*/
/*  Filename: int_routines.c           */
/*  Date: 02-12-20                     */
/*  Written by: Nercivan Kerimovska    */
/*            Anna Tomasson            */
/*=====================================*/

#include <io8515.h>
#include <ina90.h>
#include <int_routines.h>
#include <SPI_com.h>

char volume = 4;
char change_volume = 0;

char status_nbr;
char tts_status0;
char tts_status1;

char previous_received_char3;
char previous_received_char2;
char previous_received_char1;
char received_char;
char ready_to_receive;
char ready_to_convert;

/*interrupt [RESET_vect] void RESET_interrupt(void)
{
}*/
```

```
/*Interrupt routine for increasing the volume  */
interrupt [INT0_vect] void INT0_interrupt(void)
{
 change_volume = 1;
}

/*Interrupt routine for decreasing the volume  */
interrupt [INT1_vect] void INT1_interrupt(void)
{
 change_volume = 2;
}

interrupt [TIMER1_CAPT1_vect] void TIMER1_CAPT1_interrupt(void){}

interrupt [TIMER1_COMPA_vect] void TIMER1_COMPA_interrupt(void){}

interrupt [TIMER1_COMPB_vect] void TIMER1_COMPB_interrupt(void){}

interrupt [TIMER1_OVF1_vect] void TIMER1_OVF1_interrupt(void){}

interrupt [TIMER0_OVF0_vect] void TIMER0_OVF0_interrupt(void){}

/*Interrupt routine for reading status bytes  */
interrupt [SPI_STC_vect] void SPI_STC_interrupt(void)
{
 switch (status_nbr){
 case 1:
  tts_status0 = SPDR;                    //Status received after sending the first
                                         byte
  break;
 case 2:
  tts_status1 = SPDR;                    //Status received after sending the second
                                         byte
  break;
 }
 status_nbr = 0;
}

/*Interrupt routine excecuted when Received character complete (RXC in USR is set)  */
interrupt [UART_RX_vect] void UART_RX_interrupt(void)
{
 if ((USR & 0x10) == 0x00){              //FE=0 => no errors detected
  received_char = UDR;                   //The UDR register is read

  if (ready_to_receive == 1){
   chars_for_conversion[nbr_of_chars] = received_char; //The recieved characters are
placed in the vector chars_for_conversion[]
   nbr_of_chars ++;
  }

  if (previous_received_char3 == '#' && previous_received_char2 == '#' &&
previous_received_char1 == '#' && received_char != '#'){
   chars_for_conversion[0] = received_char;    //The first character received
   nbr_of_chars = 1;
   ready_to_receive = 1;
```

```
  }

  else if (previous_received_char1 == '%' && received_char == '%'){
    chars_for_conversion[--nbr_of_chars] = 0x00;
    chars_for_conversion[--nbr_of_chars] = 0x00;
    ready_to_receive = 0;
    ready_to_convert = 1;                    //All characters received an placed in the
vector
   }
   previous_received_char3 = previous_received_char2;
   previous_received_char2 = previous_received_char1;
   previous_received_char1 = received_char;
  }
  else{
   UDR;                                      // The UDR register has to be read even if
                                             FE=1
  }
}

interrupt [UART_UDRE_vect] void UART_UDRE_interrupt(void){}

interrupt [UART_TX_vect] void UART_TX_interrupt(void){}

interrupt [ANA_COMP_vect] void ANA_COMP_interrupt(void){}
```

D.1.3 SPI_com.c

```
/*=================================*/
/*  Filename: SPI_com.c            */
/*  Date: 02-12-20                 */
/*  Written by: Nercivan Kerimovska*/
/*          Anna Tomasson          */
/*=================================*/

#include <io8515.h>
#include <ina90.h>
#include <SPI_com.h>
#include <int_routines.h>

char chars_for_conversion [200];          //Vector containing the characters of the
word to be converted
char nbr_of_chars;                        //Indicates how many characters the word
                                          consits of

/* Subroutine for shifting bytes to SPDR (type 1 commands)  */
void send_proc(char cmd_byte, char cmd_data_byte);

/*Subroutine for shifting bytes to SPDR (type 3 commands)   */
void send_data(void);


/* Subroutine for sending command type 1 from processor to text-to-speech circuit   */
void send_cmd(char cmd_byte, char cmd_data_byte)
{
 do{
   send_proc(0x04, 0x00);                 //RDST = read status
```

```
 }while ((tts_status1 & 0x80) != 0x80);        //Wait until R/B = 1 (R/B = ready/busy)

 do{
  send_proc(cmd_byte, cmd_data_byte);          //send data
 }while ((tts_status1 & 0x01) == 0x01);        //Resend while ICMD = 1 (ICMD =
                                               ignore command)
}

/* Subroutine for shifting bytes to SPDR (type 1 commands)  */
void send_proc(char cmd_byte, char cmd_data_byte)
{
 PORTA |= 0x01;                                //SS\ = 1
 PORTA &= 0xFE;                                //SS\ = 0

 status_nbr = 1;
 SPDR = cmd_byte;                              //Send the first command byte
 delay();

 status_nbr = 2;
 SPDR = cmd_data_byte;                         //Send the second command byte
 delay();

 PORTA |= 0x01;                                //SS\ = 1
}

/* Subroutine for converting text to speech   */
void convert(void)
{
 do{
  send_proc(0x04, 0x00);                       //RDST = read status
 }while ((tts_status1 & 0x80) != 0x80);        //Wait until R/B = 1 (R/B = ready/busy)

 do{
  send_data();                                 //send data
 }while ((tts_status1 & 0x01) == 0x01):        //Resend while ICMD = 1 (ICMD =
                                               ignore command)

 nbr_of_chars = 0;
 send_cmd(0x4C, 0x00);                         //FIN = finish
 send_cmd(0x06, 0x00);                         //RINT = read interrupt
}

/*Subroutine for shifting bytes to SPDR (type 3 commands)   */
void send_data(void)
{
 char i;

 PORTA |= 0x01;                                //SS\ = 1
 PORTA &= 0xFE;                                //SS\ = 0

 status_nbr = 1;
 SPDR = 0x81;                                  //Send the first command byte to start
                                               conversion
```

```
delay();

status_nbr = 2;
SPDR = 0x00;                                    //Send the second command byte to start
                                                conversion
delay();

chars_for_conversion[nbr_of_chars] = 0x1A; //EOT = end of text
nbr_of_chars ++;

for (i = 0; i < nbr_of_chars; i++){
 while ((PINA & 0x02) != 0x02);                 //Wait until CNVT = 1 (CNVT =
                                                converting)
 SPDR = chars_for_conversion[i];                //Send the characters one by one
 chars_for_conversion[i] = 0x00;                //Empty the vector efter hand
 delay();
 }
 PORTA |= 0x01;                                 //SS\ = 1
}

/* Delay routine   */
void delay(void)
{
 int i;
 for(i = 1; i > 0; i--);
}
```

D.2 The H-files

D.2.1 int_routines.h

```
/*=====================================*/
/*  Filename: int_routines.h           */
/*  Date: 02-12-20                     */
/*  Written by: Nercivan Kerimovska    */
/*            Anna Tomasson            */
/*=====================================*/

#include <io8515.h>
#include <ina90.h>

extern char volume;
extern char change_volume;

extern char status_nbr;
extern char tts_status0;
extern char tts_status1;

extern char previous_received_char3;
extern char previous_received_char2;
extern char previous_received_char1;
extern char received_char;
extern char ready_to_receive;
extern char ready_to_convert;

//extern interrupt [RESET_vect] void RESET_interrupt(void);
```

```
extern interrupt [INT0_vect] void INT0_interrupt(void);
extern interrupt [INT1_vect] void INT1_interrupt(void);
extern interrupt [TIMER1_CAPT1_vect] void TIMER1_CAPT1_interrupt(void);
extern interrupt [TIMER1_COMPA_vect] void TIMER1_COMPA_interrupt(void);
extern interrupt [TIMER1_COMPB_vect] void TIMER1_COMPB_interrupt(void);
extern interrupt [TIMER1_OVF1_vect] void TIMER1_OVF1_interrupt(void);
extern interrupt [TIMER0_OVF0_vect] void TIMER0_OVF0_interrupt(void);
extern interrupt [SPI_STC_vect] void SPI_STC_interrupt(void);
extern interrupt [UART_RX_vect] void UART_RX_interrupt(void);
extern interrupt [UART_UDRE_vect] void UART_UDRE_interrupt(void);
extern interrupt [UART_TX_vect] void UART_TX_interrupt(void);
extern interrupt [ANA_COMP_vect] void ANA_COMP_interrupt(void);
```

D.2.2 SPI_com.h

```
/*====================================*/
/*  Filename: SPI_com.h               */
/*  Date: 02-12-20                    */
/*  Written by: Nercivan Kerimovska   */
/*          Anna Tomasson             */
/*====================================*/

#include <io8515.h>
#include <ina90.h>
#include <int_routines.h>

extern char chars_for_conversion [200];
extern char nbr_of_chars;

/* Subroutine for sending command type 1 from processor to text-to-speech circuit   */
extern void send_cmd(char cmd_byte, char cmd_data_byte);

/* Subroutine for converting text to speech    */
extern void convert(void);

/* Delay routine              */
extern void delay(void);
```

Appendix E – Settings

E.1 Settings for IAR Embedded Workbench

1. Open the IAR Embedded Workbench.
2. Enter *File/New*. Select *Project* and click OK.
3. A *New Project* window appears. Type filename (i.e. name the project) and create a directory to save the project in. Press *Create*.
4. A window with the selected project name appears. Select *Debug* in the *Targets* fall down menu.
5. Enter *File/New*. Select *Source/Text* and press OK.
6. A window named *Untitled1* appears. Enter *File/Save As* and save the file with the same name as the project i.e. projectname.c. This file should contain the main function.
7. Press *Project/Files*, a window named *Project Files* appears. Set it to look in the directory created earlier and select the c-file created earlier and press *Add* to add files in group. Press *Done*.
8. In the first window created (the project window) a directory named *Common Sources* has been added to the directory's *Debug* and *Release*. The c-file has been added to this directory (Common Sources).
9. Make sure that *Debug* is the selected target in the project window.
10. Select *Project/Options*, a window named *Option for Target Debug* appears. Here the settings for the compiler and linker are done. Changes are only to be done in the flaps mentioned below.

    Select category *General:*
    Target-flap: Select the microcontroller to be used in the process configuration fall down list. Select memory model in the fall down list.

    Select category *XLINK:*
    Include-flap: Press *Override Default* and select the right xcl-file. If it does not exist in the fall down list it can be downloaded from Atmels homepage and saved in the directory iccA90. Press OK.

11. Select the *Release* directory in the target list in the project window. Do the same steps as in step 10 and in the *Output*-flap select "other" in the *Format* frame. Select the right output format for the microcontroller (in this project Intel-extended was used) and press OK.
12. To compile and link the code, enter *Project* and click *Build All*. This creates a d90 file (if target Debug is selected in the project window) that is used for debugging purpose with the emulator or an a90 file (if target Release is selected in the project window) that can be flashed to the microcontroller.

E.2 Settings for AVR Studio

1. Open AVR Studio.
2. A window named *Welcome to AVR Studio* appears, press *Cancel.*
3. Enter *File/Open Object File* and a window named *Open Object File* appears.
4. Make sure that the emulator is connected both to the computer and the target and that everything is powered on.
5. Enter the directory created in IAR and in this enter *Debug/Exe* and open the d90-file. Then a notification window appears that says that a project using this object file already exists, press YES.
6. The project is now loaded and is ready to use with the emulator.
7. Make sure that the programmer is connected to both the computer and the target and that everything is powered on.
8. To flash the microcontroller, enter *Tools/AVRISP/AVRISP.* A window named *AVRISP* appears.

   Program-flap:
   Select the right device in the fall-down list. Select the a90-file as the input hex file to be flashed.

   Auto-flap:
   Select the wanted actions and press start.

In special cases and for other functionalities please refer to the information material from Atmel [2], [3] and [10]

CLAIMS

1. A device for text-to-speech conversion in a mobile telephone.

2. A device for text-to-speech conversion in a mobile telephone as described in the specification.

3. A method for text-to-speech conversion in a mobile telephone.

4. A method for text-to-speech conversion in a mobile telephone as described in the specification.

Europäisches Patentamt | European Patent Office | Office européen des brevets

Bescheinigung | Certificate | Attestation

Die angehefteten Unterlagen stimmen mit der ursprünglich eingereichten Fassung der auf dem nächsten Blatt bezeichneten europäischen Patentanmeldung überein.

The attached documents are exact copies of the European patent application described on the following page, as originally filed.

Les documents fixés à cette attestation sont conformes à la version initialement déposée de la demande de brevet européen spécifiée à la page suivante.

Patentanmeldung Nr. | Patent application No. | Demande de brevet n°

03011580.2 / EP03011580

The organization code and number of your priority application, to be used for filing abroad under the Paris Convention, is EP03011580

Der Präsident des Europäischen Patentamts;
Im Auftrag

For the President of the European Patent Office

Le President de l'Office européen des brevets
p.o.

R.C. van Dijk

Anmeldung Nr:
Application no.: 03011580.2
Demande no:

Anmeldetag:
Date of filing: 22.05.03
Date de dépôt:

Anmelder/Applicant(s)/Demandeur(s):

Sony Ericsson Mobile Communications AB
221 88 Lund/SE

Bezeichnung der Erfindung/Title of the invention/Titre de l'invention:
(Falls die Bezeichnung der Erfindung nicht angegeben ist, siehe Beschreibung.
If no title is shown please refer to the description.
Si aucun titre n'est indiqué se referer à la description.)

Device for generating speech, apparatus connectable to or incorporating such a device, and computer program products therefor

In anspruch genommene Priorität(en) / Priority(ies) claimed / Priorité(s) revendiquée(s)
Staat/Tag/Aktenzeichen / State/Date/File no. / Pays/Date/Numéro de dépôt:

EP / 16.12.02 / EPA 02445177

Internationale Patentklassifikation / International Patent Classification / Classification internationale de brevets:

G06F3/00

Am Anmeldetag benannte Vertragstaaten / Contracting states designated at date of filing / Etats contractants désignées lors du dépôt:

AT BE BG CH CY CZ DE DK EE ES FI FR GB GR HU IE IT LI LU MC NL PT RO SE SI SK TR

DEVICE FOR GENERATING SPEECH, APPARATUS CONNECTABLE TO OR INCORPORATING SUCH A DEVICE, AND COMPUTER PROGRAM PRODUCT THEREFOR

Field of invention

The present invention relates to a device for generating speech associated with information shown on a display, especially displays on portable devices such as mobile telephones and the like. A conversion circuit converts the data shown to audible speech helping the user to operate the apparatus. The invention also relates to an apparatus arranged to cooperate with such a device or incorporating such a device, and a computer program product therefor.

State of the art

In portable devices such as mobile telephones etc. the displays are used to display menus controlling the operation and settings of the device or other information relating to messages or games. The displays are often small, which may be a problem for the user, especially if he is visually impaired. Also for other reasons, there is a need for an audible version of the display.

The present invention solves this problem by transforming the information displayed to audible speech.

Summary of the invention

In a first aspect, the invention provides a device for generating speech, wherein a microcontroller is connectable to an apparatus for receiving data to be converted to speech, and sending the data to a conversion circuit; and a conversion circuit connectable to a speaker system for converting the data to a speech signal.

Preferably, the data is supplied as ASCII characters.

Suitably, the conversion circuit supports various selectable languages and the conversion circuit is capable of downloading languages via the connected apparatus.

Suitably, the conversion circuit supports various selectable voices and the conversion circuit is capable of downloading voices via the connected apparatus.

Preferably, the speed of the speech signal is adjustable.

Preferably, the microcontroller is connectable to a memory containing language information, such as various languages, abbreviation lists and dictionaries.

Preferably, the microcontroller is connectable to a memory containing voice settings.

Suitably, the microcontroller is connectable to the apparatus by means of a system connector having an interface for audio signals, serial channels, power leads and analog and digital ground leads.

The device may be implemented as a functional cover, comprising a shell covering the front of the apparatus and a microprocessor cooperating with the processor of the apparatus.

The connectable apparatus may be a portable telephone, a pager, a communicator or an electronic organiser.

In a second aspect, the invention provides an apparatus having a display for showing various readable data, wherein a control unit is arranged to extract readable data for sending to a device for generating speech as mentioned above.

The readable data may include texts from menus, text messages, help information, calendars or confirmation of actions taken with the apparatus.

Suitably, the control unit is arranged to extract a part of the readable data, such as a line or a word, at a time from the display and sending it automatically to the speech generating device at a fixed or controllable rate, and/or the control unit is arranged to extract a line at a time from the display and sending it to the speech generating device in dependence of scrolling in the display.

Suitably, the control unit is also arranged to extract a part of the readable data, such as a character, a line or a word, at a time from the display and sending it to the speech generating device in dependence of inputting characters to the apparatus.

Then, the control unit may be arranged to send readable data as triggered by the input of definite characters, such as letters, signs, spaces or punctuation marks.

Preferably, the control unit is arranged to extract readable data from a selected file and sending it automatically to the speech generating device at a fixed or controllable rate.

In a third aspect, the invention provides an apparatus having a display for showing various readable data, including a control unit and a device for generating speech comprising a conversion circuit for converting data to a speech signal and connectable to a speaker system, wherein the control unit is arranged to extract readable data for sending to the speech generating device.

The speaker system may be integrated with the apparatus.

Suitably, the data is supplied as ASCII characters.

Suitably, the conversion circuit supports various selectable languages, and is capable of downloading languages.

Suitably, the conversion circuit supports various selectable voices, and is capable of downloading voices.

Preferably, the speed of the speech signal is adjustable.

Suitably, the apparatus is connectable to a memory containing language information, such as various languages, abbreviation lists and dictionaries.

Suitably, the apparatus is connectable to a memory containing voice settings.

Preferably, the readable data includes texts from menus, text messages, help information, calendars or confirmation of actions taken with the apparatus.

Suitably, the control unit is arranged to extract a part of the readable data, such as a line or a word, at a time from the display and sending it automatically to the speech generating device at a fixed or controllable rate, and/or the control unit is arranged to extract a line at a time from the display and sending it to the speech generating device in dependence of scrolling in the display.

Suitably, the control unit is arranged to extract a part of the readable data, such as a character, a line or a word, at a time from the display and sending it to the speech generating device in dependence of inputting characters to the apparatus.

Then, the control unit may be arranged to send readable data as triggered by the input of definite characters, such as letters, signs, spaces or punctuation marks.

Preferably, the control unit is arranged to extract readable data from a selected file and sending it automatically to the speech generating device at a fixed or controllable rate.

The apparatus may be a portable telephone, a pager, a communicator or an electronic organiser.

In a fourth aspect, the invention provides a computer program product loadable into the internal memory of an apparatus having a display for showing various readable data, wherein the computer program product comprises software code portions to achieve the functionality of the apparatus as mentioned above.

The computer program product may be embodied on a computer readable medium.

Brief description of the drawings

Embodiments of the invention will be described in detail below with reference to the accompanying drawings, of which:

fig. 1 is a block diagram of the main blocks of the invention,

fig. 2 is a perspective view of a system connector,

fig. 3 is a data flow diagram, and

fig. 4 is an example of a mobile phone using the present invention.

Detailed description of preferred embodiments

The invention will be described in relation to a mobile phone including text-to-speech conversion. The invention is also applicable in many other devices, e.g. pagers, communicators, electronic organisers and the like portable devices.

Text-to-speech conversion is a feature that is of interest in many different areas and applications. One of the more interesting is the use in mobile phones. Today mobile phones are used by almost everyone and a feature like this can be an important aid, especially for the visually impaired and for users who need to focus on other things while using the phone, for instance car drivers using hands-free equipment. The text-to-speech conversion is done in hardware with a text-to-speech circuit. A highlighted menu label, an SMS or other readable data are sent to a microcontroller. The data may be received as ASCII characters and these are forwarded to the text-to-speech circuit by the microcontroller. The text-to-speech circuit converts the characters to audio signals and sends them to a loudspeaker system.

The invention makes the mobile telephone more user-friendly by reading messages and menus to help the user locate himself while browsing the menus system.

Fig. 1 shows an embodiment of the invention in which the speech generating device is implemented as an accessory. The accessory is to be attached to a mobile phone 1 via its system connector. The accessory may be implemented as a so called active or functional cover, that is a shell covering e.g. the front of the phone and also connected to the phone's system connector. The functional cover contains a microprocessor holding additional functions and cooperating with the processor of the telephone. Thus, the actual outer shape of the accessory depends on the mobile phone and is not shown here.

The speech generating device 5 is shown within the dashed square and includes a microcontroller 6 receiving the data to be converted from the mobile phone and passing it to a text-to-speech (TTS) circuit 7. The TTS circuit 7 converts the text to audio signals and sends them via an (optional) amplifier 8 to a loudspeaker 9.

In another embodiment, the speech generating device is built into the mobile phone and may use the internal hardware, software and speaker system 11, see figure 4. Existing telephones are usually provided with a microprocessor and a digital signal processor capable of being programmed to perform the required text to speech conversion. Thus, the text to speech conversion may be embodied as a software product, e.g. a computer program on a readable medium or deliverable through the Internet.

The microcontroller may for example be a commercially available circuit comprising a programmable flash memory, general purpose input/output lines and working registers, internal and external interrupts, a programmable serial universal asynchronous receiver and transmitter (UART) and a port for a serial peripheral interface. The registers are programmed to control the behaviour of the microcontroller in the desired way. The microcontroller is responsible for receiving the data to be converted to speech and sending the data to the TTS circuit.

The TTS circuit 7 may be a commercially available circuit. The circuit should have an output designed to drive a speaker, and preferably also a telesocket for headphone or an external loudspeaker. To get a higher volume a general amplifier 8 could be used, e.g. a fully differential audio power amplifier.

The TTS circuit should also support SMS (Short Message Service) and preferably a modifiable abbreviation list. The TTS circuit also should support various languages. In a preferred embodiment it is possible to program other languages through a serial port allowing the user to download different languages. A standard speaker voice is built-in, but preferably it is also possible to download

different speaker voices or connect external memories, for instance so called memory sticks, containing voice data. When the speech generating device is connected or integrated in a mobile phone or communicator, databases could be downloaded via the telecommunication network or the Internet.

The TTS circuit receives data to be read through its input port, e.g. ASCII characters, converts it into spoken audio and sends it to an analog output. A typical circuit comprises a text processor, a smoothing filter and multilevel memory storage array. The voice and audio signals are stored in the memory in their natural, uncompressed form, which provides a good voice reproduction quality.

The speech conversion is conventional and is not described in detail here. Briefly, the text-to-speech mechanism comprises text normalisation, word to phoneme conversion and phoneme mapping. The text normalisation is the process of translating the incoming text to pronounceable words. It expands abbreviations and translates numeric strings to spoken words. The abbreviation list can be modified. This enables flexibility of adding abbreviations specifically for the text, either by the developer or by the end user to customise the device. Even the unique characters of SMS are supported, meaning that icons such as smilies ;-) will be replaced by its corresponding true spoken meaning. This means that an SMS containing abbreviations and icons will be correctly recited.

The TTS circuit should have an internal input buffer that could hold at least 256 characters in order to receive an entire SMS consisting of 160 characters. This means that no extra memory is needed in the connecting apparatus.

The microcontroller 6 preferably is connected to a volume control to adjust the volume of a speaker system connected. For instance, two buttons could be provided, one to increase the volume and one to decrease the volume. The buttons are suitably connected to the interrupt pins of the microcontroller.

The speech generating device is provided with an interface for connecting the device to the phone via its system connector. The system connector interface comprises audio signals, two serial channels, power leads and the analog and digital ground leads. A typical system connector interface 10 is shown in fig. 2.

The mobile telephone is arranged to extract texts and characters from the data shown on the display and to send it to the speech generating device. The extracted text string may be sent to the device to place the data on the system bus. All text strings are stored in a list and a text ID is a pointer used to point out the different text strings.

Fig. 3 shows the data flow diagram between the blocks in the system. The different blocks need the right interfaces to communicate properly with each other. The interface between the phone 1 and the microcontroller 6 consists of a universal asynchronous receiver and transmitter UART, while the microcontroller 6 and the

TTS circuit 7 communicate via a serial peripheral interface. The UART may form part of a commercial microcontroller.

Fig. 4 shows an example of the operation of the present invention. The mobile phone 1 includes a display 2 currently showing part of a message, e.g. an SMS. The keypad includes scroll buttons 3 for moving in the display. Currently one line 4 of the display is marked by highlighting the text. In an automatic mode, the control unit extracts one line or word after another at a fixed or adjustable rate and sends it automatically to the speech generating device for translating into spoken audio signals. It is preferably possible to pause, rewind and move fast forward in the text. The speed of the speech reading the text can be adjusted to suit each individual.

In another mode, the user scrolls in the display by means of the buttons 3 to select one line for sending to conversion circuit and reading aloud. The user may also select a whole text or a file, such as a message or downloaded article. The selected text is sent to the conversion circuit.

In a further mode, the text to speech conversion is active when the user is writing a message, such as an SMS. After inputting a letter or sign, this is read aloud. When a whole word is finished, e.g. as triggered by the input of a space, the word is sent to the conversion circuit and read aloud. Further, when a punctuation mark is input the whole last sentence may be read, and finally the whole message may be read before it is sent. The control unit sends the text to be read automatically in dependence of a definite set of characters, such as spaces and punctuation marks, and also, optionally, each input sign or letter.

The text-to-speech conversion in the phone is not only an aid for the visually impaired and car drivers but also a step further in personalising the phone. Some of the possibilities with the text-to-speech function in a mobile telephone are:

- Interaction with voice control. A voice command from the user can be used to control functions in the phone, like make a call or navigating in menus, and the speech function can then confirm the commands and possibly add help messages.
- Extended help functions, giving spoken explanations to a selected topic, like a step-by-step instruction on how to install an e-mail account. The whole instruction manual can be accessed in this way. This function can be activated and controlled by a shortcut or by voice recognition.
- By saving texts on memory sticks connectable to the device or the mobile phone, it is possible to have huge text masses like books read.
- Reading reminder and alerts from a calendar.
- Reading pages and articles downloaded from the Internet or by WAP.

- Use as a navigation aid together with GPS (Global Positioning System) and the Yellow Pages route service.

Different voices are possible. It is contemplated that popular voices like film stars etc. could be available for downloading or sold as connectable memory sticks. The spoken audio signal could also be combined with music files, e.g. MIDI (Musical Instrument Digital Interface) files.

The invention may be implemented as a separate accessory connectable to an apparatus, or an apparatus incorporating such a device. The invention also relates to an apparatus connectable to such a device. The invention may be implemented by hardware or by software included in a self-contained apparatus or various combinations thereof. The scope of the invention is only limited by the claims below.

CLAIMS

1. A device (5) for generating speech, **characterised** by:
a microcontroller (6) connectable to an apparatus for receiving data to be converted to speech, and sending the data to a conversion circuit (7);
a conversion circuit (7) connectable to a speaker system (9) for converting the data to a speech signal.

2. A device according to claim 1, **characterised** in that the data is supplied as ASCII characters.

3. A device according to claim 1 or 2, **characterised** in that the conversion circuit (7) supports various selectable languages.

4. A device according to claim 3, **characterised** in that the conversion circuit (7) is capable of downloading languages via the connected apparatus.

5. A device according to any one of claims 1 to 4, **characterised** in that the conversion circuit (7) supports various selectable voices.

6. A device according to claim 5, **characterised** in that the conversion circuit (7) is capable of downloading voices via the connected apparatus (1).

7. A device according to any one of claims 1 to 6, **characterised** in that the speed of the speech signal is adjustable.

8. A device according to any one of claims 1 to 7, **characterised** in that the microcontroller (6) is connectable to a memory containing language information, such as various languages, abbreviation lists and dictionaries.

9. A device according to any one of claims 1 to 8, **characterised** in that the microcontroller (6) is connectable to a memory containing voice settings.

10. A device according to any one of claims 1 to 9, **characterised** in that the microcontroller (6) is connectable to the apparatus (1) by means of a system connector having an interface (10) for audio signals, serial channels, power leads and analog and digital ground leads.

11. A device according to claims 10, **characterised** in that the device is implemented as a functional cover, comprising a shell covering the front of the

apparatus (1) and a microprocessor cooperating with the processor of the apparatus (1).

12. A device according to any one of claims 1 to 11, **characterised** in that the connectable apparatus (1) is a portable telephone, a pager, a communicator or an electronic organiser.

13. An apparatus (1) having a display (2) for showing various readable data, **characterised** by a control unit arranged to extract readable data for sending to a device (5) for generating speech in accordance with any one of the preceding claims.

14. An apparatus according to claim 13, **characterised** in that the readable data includes texts from menus, text messages, help information, calendars or confirmation of actions taken with the apparatus (1).

15. An apparatus according to claims 13 or 14, **characterised** in that the control unit is arranged to extract a part of the readable data, such as a line or a word, at a time from the display (2) and sending it automatically to the speech generating device (5) at a fixed or controllable rate.

16. An apparatus according to claims 13, 14 or 15, **characterised** in that the control unit is arranged to extract a part of the readable data, such as a line or a word, at a time from the display (2) and sending it to the speech generating device (5) in dependence of scrolling in the display (2).

17. An apparatus according to claims 13, 14, 15 or 16, **characterised** in that the control unit is arranged to extract a part of the readable data, such as a line or a word or a character, at a time from the display (2) and sending it to the speech generating device (5) in dependence of inputting characters to the apparatus.

18. An apparatus according to claims 17, **characterised** in that the control unit is arranged to send readable data as triggered by the input of definite characters, such as letters, signs, spaces or punctuation marks.

19. An apparatus according to any one of claims 13 to 18, **characterised** in that the control unit is arranged to extract readable data from a selected file and sending it automatically to the speech generating device (5) at a fixed or controllable rate.

20. An apparatus (1) having a display for showing various readable data, **characterised** by including a control unit and a device for generating speech comprising a conversion circuit for converting data to a speech signal and connectable to a speaker system (9; 11), wherein the control unit is arranged to extract readable data for sending to the speech generating device .

21. An apparatus according to claim 20, **characterised** in that the speaker system (11) is integrated with the apparatus.

22. An apparatus according to claim 20 or 21, **characterised** in that the data is supplied as ASCII characters.

23. An apparatus according to claim 20, 21 or 22, **characterised** in that the conversion circuit supports various selectable languages.

24. An apparatus according to claim 23, **characterised** in that the apparatus (1) is capable of downloading languages.

25. An apparatus according to any one of claims 20 to 24, **characterised** in that the conversion circuit supports various selectable voices.

26. An apparatus according to claim 25, **characterised** in that the apparatus (1) is capable of downloading voices.

27. An apparatus according to any one of claims 206 to 26, **characterised** in that the speed of the speech signal is adjustable.

28. An apparatus according to any one of claims 20 to 27, **characterised** in that the apparatus (1) is connectable to a memory containing language information, such as various languages, abbreviation lists and dictionaries.

29. An apparatus according to any one of claims 20 to 28, **characterised** in that the apparatus (1) is connectable to a memory containing voice settings.

30. An apparatus according to any one of claims 20 to 29, **characterised** in that the readable data includes texts from menus, text messages, help information, calendars or confirmation of actions taken with the apparatus (1).

31. An apparatus according to any one of claims 20 to 29, **characterised** in that the control unit is arranged to extract a part of the readable data, such as a line or a word, at a time from the display and sending it automatically to the speech generating device at a fixed or controllable rate.

32. An apparatus according to any one of claims 20 to 31, **characterised** in that the control unit is arranged to extract a part of the readable data, such as a line or a word, at a time from the display and sending it to the speech generating device in dependence of scrolling in the display (2).

33. An apparatus according to any one of claims 20 to 32, **characterised** in that the control unit is arranged to extract a part of the readable data, such as a character, a line or a word, at a time from the display (2) and sending it to the speech generating device (5) in dependence of inputting characters to the apparatus.

34. An apparatus according to claims 33, **characterised** in that the control unit is arranged to send readable data as triggered by the input of definite characters, such as letters, signs, spaces or punctuation marks.

35. An apparatus according to any one of claims 20 to 34, **characterised** in that the control unit is arranged to extract readable data from a selected file and sending it automatically to the speech generating device (5) at a fixed or controllable rate.

36. An apparatus according to any one of claims 13 to 35, **characterised** in that the apparatus is a portable telephone, a pager, a communicator or an electronic organiser.

37. A computer program product loadable into the internal memory of an apparatus (1) having a display for showing various readable data, **characterised** by comprising software code portions to achieve the functionality of the apparatus in accordance with any one of claims 20 to 36.

38. A computer program product according to claim 37, embodied on a computer readable medium.

ABSTRACT

The invention relates to a device for generating speech associated with information shown on a display (2), especially displays on portable devices such as mobile telephones (1) and the like. A conversion circuit converts the data shown to audible speech helping the user to operate the apparatus. The invention also relates to an apparatus arranged to cooperate with such a device or incorporating such a device, and a computer program product therefor.

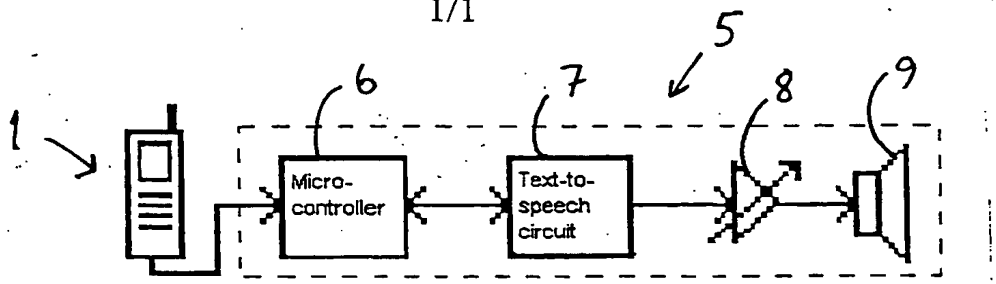


FIG 1

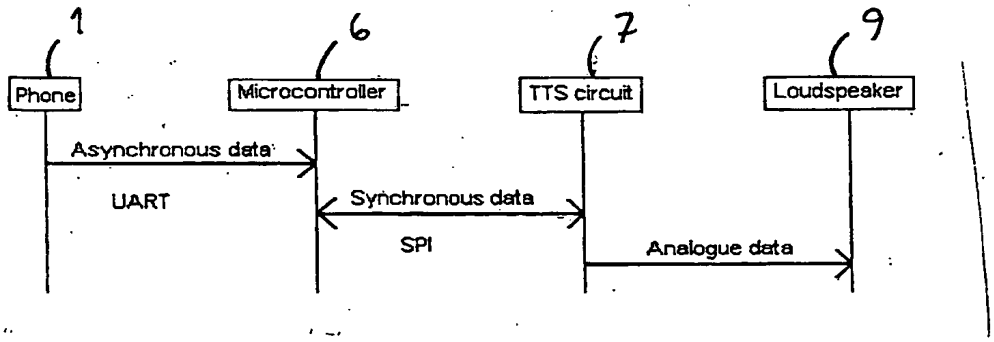


FIG 3

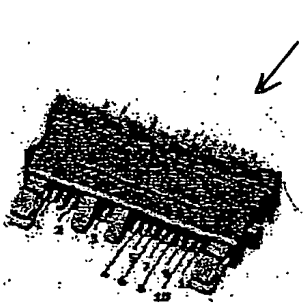


FIG 2

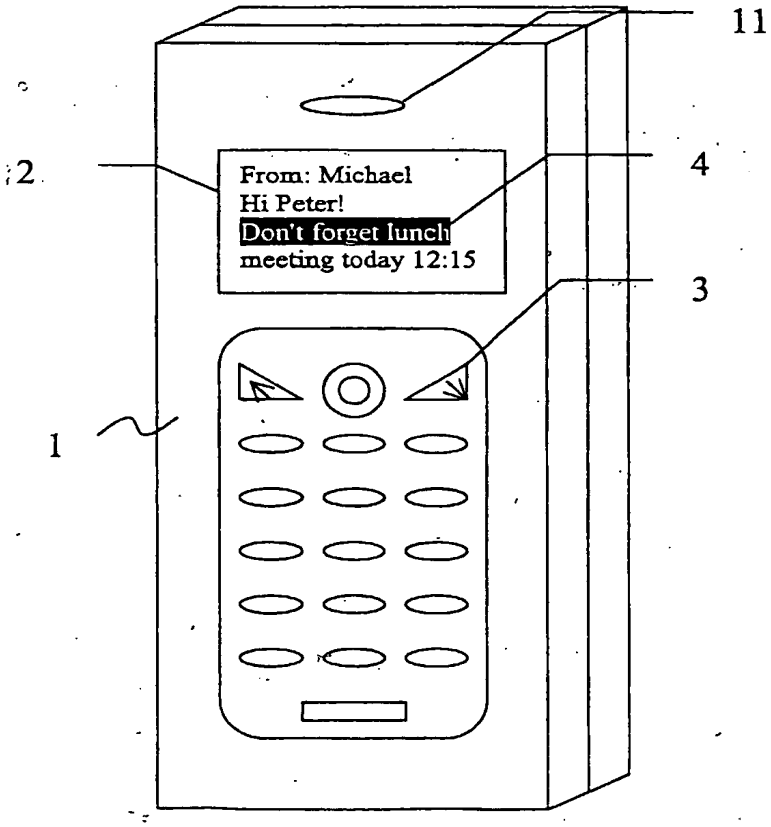


FIG 4

Europäisches Patentamt | European Patent Office | Office européen des brevets

Bescheinigung | Certificate | Attestation

Die angehefteten Unterlagen stimmen mit der ursprünglich eingereichten Fassung der auf dem nächsten Blatt bezeichneten europäischen Patentanmeldung überein.

The attached documents are exact copies of the European patent application described on the following page, as originally filed.

Les documents fixés à cette attestation sont conformes à la version initialement déposée de la demande de brevet européen spécifiée à la page suivante.

Patentanmeldung Nr. | **Patent application No.** | **Demande de brevet n°**

02445177.5 / EP02445177

The organization code and number of your priority application, to be used for filing abroad under the Paris Convention, is EP02445177

Der Präsident des Europäischen Patentamts;
Im Auftrag

For the President of the European Patent Office

Le President de l'Office européen des brevets
p.o.

R.C. van Dijk

KV83387

Anmeldung Nr:
Application no.: 02445177.5
Demande no:

Anmeldetag:
Date of filing: 16.12.02
Date de dépôt:

Anmelder/Applicant(s)/Demandeur(s):

Sony Ericsson Mobile Communications AB
221 88 Lund/SE

Bezeichnung der Erfindung/Title of the invention/Titre de l'invention:
(Falls die Bezeichnung der Erfindung nicht angegeben ist, siehe Beschreibung.
If no title is shown please refer to the description.
Si aucun titre n'est indiqué se referer à la description.)

Text to speech help in a mobile phone

In anspruch genommene Priorität(en) / Priority(ies) claimed / Priorité(s) revendiquée(s)
Staat/Tag/Aktenzeichen / State/Date/File no. / Pays/Date/Numéro de dépôt:

Internationale Patentklassifikation / International Patent Classification / Classification internationale de brevets:

G01L13/00

Am Anmeldetag benannte Vertragstaaten / Contracting states designated at date of filing / Etats contractants désignées lors du dépôt:

AT BE BG CH CY CZ DE DK EE ES FI FR GB GR IE IT LI LU MC NL PT SE SI SK TR